Robust Learning Architectures for Perceiving Object Semantics and Geometry

by

Chi Li

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

April, 2018

Abstract

Parsing object semantics and geometry in a scene is one core task in visual understanding. This includes classification of object identity and category, localizing and segmenting an object from cluttered background, estimating object orientation and parsing 3D shape structures. With the emergence of deep convolutional architectures in recent years, substantial progress has been made towards learning scalable image representation for large-scale vision problems such as image classification. However, there still remains some fundamental challenges in learning robust object representation. First, creating object representations that are robust to changes in viewpoint while capturing local visual details continues to be a problem. In particular, recent convolutional architectures employ spatial pooling to achieve scale and shift invariances, but they are still sensitive to out-of-plane rotations. Second, deep Convolutional Neural Networks (CNNs) are purely driven by data and predominantly pose the scene interpretation problem as an end-to-end black-box mapping. However, decades of work on perceptual organization in both human and machine vision suggests that there are often intermediate representations that are intrinsic to an inference task, and which provide essential structure to improve generalization.

In this dissertation, we present two methodologies to surmount the aforementioned two issues. We first introduce a multi-domain pooling framework which group local visual signals within generic feature spaces that are invariant to 3D object transformation, thereby reducing the sensitivity of output feature to spatial deformations. We formulate a probabilistic analysis of pooling which further suggests the multi-domain pooling principle. In addition, this principle guides us in designing convolutional architectures which achieve state-of-the-art performance on instance classification and semantic segmentation. We also present a multi-view fusion algorithm which efficiently computes multi-domain pooling feature on incrementally reconstructed scenes and aggregates semantic confidence to boost long-term performance for semantic segmentation.

Next, we explore an approach for injecting prior domain structure into neural network training, which leads a CNN to recover a sequence of intermediate milestones towards the final goal. Our approach supervises hidden layers of a CNN with intermediate concepts that normally are not observed in practice. We formulate a probabilistic framework which formalizes these notions and predicts improved generalization via this deep supervision method. One advantage of this approach is that we are able to generalize the model trained from synthetic CAD renderings of cluttered scenes, where concept values can be extracted, to real image domain. We implement this deep supervision framework with a novel CNN architecture which is trained on synthetic image only and achieves the state-of-the-art performance of 2D/3D keypoint localization on real image benchmarks.

Finally, the proposed deep supervision scheme also motivates an approach for accurately inferring six Degree-of-Freedom (6-DoF) pose for a large number of object classes from single or multiple views. To learn discriminative pose features, we integrate three new capabilities into a deep CNN: an inference scheme that combines both classification and pose regression based on an uniform tessellation of SE(3), fusion of a class prior into the training process via a tiled class map, and an additional regularization using deep supervision with an object mask. Further, an efficient multi-view framework is formulated to address single-view ambiguity. We show the proposed multi-view scheme consistently improves the performance of the single-view network. Our approach achieves the competitive or superior performance over the current state-of-the-art methods on three large-scale benchmarks.

Primary Readers

Gregory D. Hager (Primary Advisor)
Mandell Bellmore Professor
Department of Computer Science
Johns Hopkins University

Alan Yuille
Bloomberg Distinguished Professor
Department of Computer Science and Cognitive Science
Johns Hopkins University

Austin Reiter
Research Assistant Professor
Department of Computer Science
Johns Hopkins University

Acknowledgments

First of all, I would like to give special thanks to my primary advisor Professor Greg Hager. Greg has given me constant support and encouragement for my research, which makes my PhD career such a memorable, pleasant and fruitful journey in my life. He taught me how to cast an idea to a big picture, how to view a problem from a different perspective and how to convey an idea in an elegant and concise way. More importantly, I learned many life lessons from him about scheduling a balanced life, valuing work partners, respecting ideas and suggestions from others, and being faithful to what you believe and what you love. I am very grateful to have Greg as my advisor who really raise my mindset to a new horizon and inspires me to explore a more productive and sustainable life in the future.

Next, I would like to thank Professor Austin Reiter who joined Hopkins at the time of my most struggling period. Austin was always responsive, kind and supportive. When I first came up with the idea of multi-domain pooling, no one except Austin appreciates it and firmly encourages me to conduct experiments. He plays the critical role in building up my courage and confidence in solving hard problems. In addition, I would like to thank Professor Alan Yuille for being one of the co-advisors and committee members. Alan

deeply impressed me with his devotion, passion, love for scientific research. I learned from him to view this world from a statistical perspective. Finally, I want to thank Professor Rene Vidal, Carey Priebe and Misha Kazhdan for testing me on my GBO exam, and thank Professor Nassir Navab and Federico Tombari for their great help on my work of incremental scene understanding.

Early in my undergraduate study, I was lucky to be advised by Le Lu and Professor Hanzi Wang. They stimulated my interest in computer vision research and guided me with much patience and passion. Thank them for appreciating my potential and recommending me to Greg, which made everything happen.

I am also very grateful that I can grow up with a group of smart, nice and diligent lab colleagues including Colin, Chris, Jon, Kel, Chong, Jin, Anand, Narges, TK, Rob, Molly, Ayushi, Jonathan, Qianli, Yongsu, Kai, Purnima, Ioana, Yixin, Yan, Andrew, Amanda and Felix. I did enjoy interacting with all these guys who could often inspire me to come up with new ideas. I also want to thank some master and undergrad students (Hanyue, Bo, Yuanwei, Eric Carlson and Han) who helped me create JHU Perception datasets.

I also greatly benefited from my three internships with Apple, Microsoft Research and NEC Laboratories America. I would like to acknowledge the great research and practical training I obtained from my internship mentors: Zeeshan Zia, Manmohan Chandraker, Zhengyou Zhang, Zicheng Liu, Yinpeng Chen, Xiang Yu, Quoc-Huy Tran, Weiyu Zhang and Fredrik Larsson. They trained me to be a better researcher who values practical impact in real

life. More importantly, I got chance to work on many different vision problems, which really improves my problem-solving skills in different contexts and motivates me to develop a deeper understanding towards fundamental problems in vision and learning.

Also, thank every staff member in LCSR and CS department. This list is obviously too long to enumerate but I still want to note great help from Tracy Marshall, Zach Burnwell, Cathy Thornton, Debbie Deford, Alison Morrow, Laura Graham and Rose Chase. Their solid work did make my life in Hopkins much easier as an international student.

At last, I would like to thank myself for being faithful to my soul, fighting for my belief without doubt, and keeping confidence on my work and life.

Dedications

This dissertation is first dedicated to my wife Anni Zhou. Because of her continued support, love and encouragement throughout my PhD study and also within my whole life, she has helped me more ways than anyone else and being my long-lasting soul accompany even if we are physically separated for many years. Also, thank my parents, for all the care, love, education and comfort they have given me since I was born. Last, to my whole family, present and future. Without you, I cannot be what I am today.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

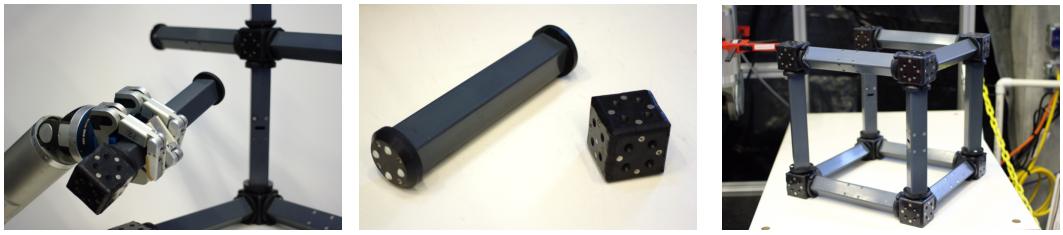
Object semantics and geometry are two fundamental elements to recognize and reason about the world. On the one hand, object semantics includes the identity, category, location and segmentation of objects in a scene. These properties depict the layout of the world and capture the underlying relationship between objects. Object geometry, on the other hand, involves the object orientation, 6-DoF pose and 2D/3D shape structures. It describes the scene occupancy, implies object functionality and encodes underlying physics.

To interact with the surrounding environment, many artificial systems are required to be capable of parsing both semantics and geometry in a scene. In the context of robotic manipulation, a visual perception system should equip a robot to detect objects of interest, estimate their poses and plan grasping based on their 3D geometries. To operate hand tools as human does, such a visual component should further localize key parts of objects that afford task specific functionalities so that the robot could arrange the appropriate motion trajectories for those parts to complete a task. For an autonomous driving car, knowing the orientations of other cars and pedestrians serves to predict the

future motions and locations of other subjects and avoid the collision.

With the emergence of deep neural network in the recent years, substantial progress has been made in learning appropriate representations for estimating object semantics and geometry, as we will discuss it in Chapter 2. However, there still remains some fundamental problems untouched. First, current state-of-the-art image features are still sensitive to 3D transformation or out-of-plane rotation due to the spatial pooling. This could cause generalization failure of identifying the object if the training data only covers limited viewpoints. Second, the deep architecture, as an end-to-end trainable machine, is solely data-driven and ignores the underlying reasoning mechanism for an inference task. Therefore, overfitting to training data frequently occurs because the model tends to "remember" the data pattern but not "reason" about the underlying rules.

In this dissertation, we present two methodologies to address the two aforementioned problems in Chapter 3 and Chapter 4, respectively. Further, these two methodologies offer the guidelines to design new types of deep architectures to generate robust representations for various vision applications including object instance classification, semantic segmentation, object pose estimation and 3D structure prediction. Additionally, we go beyond the single-view scenario and explore how to improve the recognition performance of object pose estimation via multiple views from a video sequence in Chapter 5. This is particularly helpful in the context of robotics applications where continuous video streams are available and different viewpoints could compensate for each other to resolve the heavy occlusion in cluttered scenes.



(a) Robotic assembly **(b)** Atomic objects **(c)** Lattice structure

Figure 1.1: In a robotic assembly scenario (1.1a), for the objects (1.1b) that have little or no distinguishing texture features, a robot may aim to construct a complicated lattice structure (1.1c).

1.1 Motivating Scenarios

In this section, we first discuss one compelling scenario in robotics where object semantics and geometry are both necessities to drive a robot to complete a manipulation task. Subsequently, we show how autonomous driving technology can benefit from understanding the pose and geometry of objects. Last, we discuss several fundamental limitations on single-view perception and motivate the multi-view recognition framework.

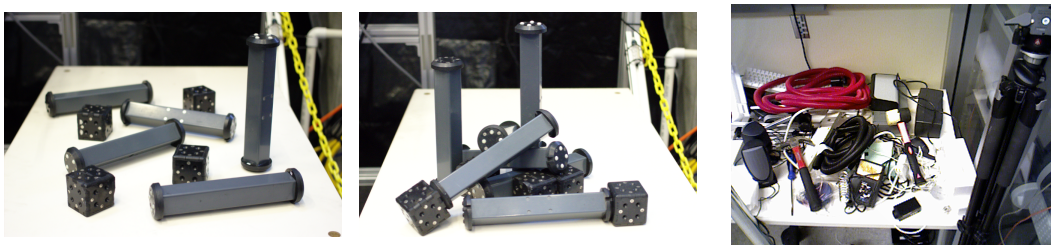
1.1.1 Robotic Manipulation

Object manipulation is one fundamental skill for robots to interact with the world. In many industrial automation domains, we may face an assembly task in which a robot is required to construct structures from rigid components which have no discriminative texture, as seen in Figure 1.1a. The lattice structures like the one shown in Figure 1.1c are built out of truss-like "links" which are joined together with coupling "nodes" via gendered magnetic surfaces, as seen in Figure 1.1b. While these components were originally designed for

open-loop assembly via quadcopter robots [1], their mechanical properties make them ideal for autonomous and semi-autonomous [2] manipulation in assembly.

To achieve the assembly task above, it is required to precisely localize objects and estimate their 3D poses since errors can result in costly failures of manipulation. Unfortunately, many object registration algorithms [3, 4, 5, 6] are developed to perform well only in "partially cluttered" scenes where individual objects are well-separated like that shown in Figure 1.2a. Furthermore, despite the substantial progress made in deep learning for image classification over the past few years, few existing deep architectures are designed to reliably detect and estimate the poses of objects once they have been piled into cluttered scenes as shown in Figure 1.2b and/or within cluttered background 1.2c in general settings.

Subsequently, roboticists are either forced to redesign tasks to accommodate the capabilities of the available object recognition algorithms, or they need to modify the objects used in a task for easier recognition. Modifications to the objects usually involve adding easily-classifiable colors, artificial texture, or easily-recognizable artificial planar markers or marker constellations [7, 8]. Unfortunately, such modifications are often impractical and sometimes even infeasible – for example, in manufacturing and assembly applications, robotic search-and-rescue, and any operation in hazardous or extreme environments. Even if the application allowed for it, augmenting the parts with 2D planar markers is still insufficient for precise pose estimation due to the small size of the parts and the range at which they need to be observed.



(a) Well separated **(b)** Densely cluttered **(c)** Cluttered background

Figure 1.2: Many object recognition systems would rely on objects being well-separated (1.2a) and fail when objects are densely packed (1.2b) and in cluttered background (1.2c).

1.1.2 Autonomous Driving

Autonomous driving technologies aim to offer the capabilities of sensing and reacting the surrounding environments without human intervention. One of the key modules in such technologies is to enable a reliable perception system which provides various information about road objects. This incorporates many vision techniques including object detection, pose estimation and 3D object structure inference. In particular, self-driving cars need to parse the orientations of objects for correct decision making and future prediction. For example, traffic lights at intersections are applied to different lanes while being spatially contingent, as shown in Figure 1.3a. Besides knowing the location and types of traffic lights (red, green or yellow signals), a self-driving car also requires the orientations of traffic lights to determine which traffic light is guiding the current lane and take appropriate actions based on these perception results. In addition, correctly predicting the orientations of pedestrians and cars in crowd street scenes, as shown in Figure 1.3b and 1.3c, offers the ways to infer the future activities of pedestrians and cars. Finally, beyond the object orientation, understanding 3D structures of objects and scenes enable

(a) Traffic Light **(b)** Pedestrian **(c)** Car

Figure 1.3: Orientations of driving-related objects such as traffic light (1.3a), car (1.3c) and pedestrian (1.3b) are keys for decision making, scene understanding and future prediction.

the self-driving cars to perceive physical free space in environment and figure out the occlusion patterns between objects in 3D. The aforementioned vision technologies together assist self-driving cars to reason future world states and plan optimal navigation paths to avoid collision.

1.1.3 Single-View vs. Multi-View Perception

Over the past few years, substantial progress has been made by deep learning methods for single-view object classification [9, 10, 11, 12, 13], semantic segmentation [14, 13, 15, 12], and object pose estimation [16, 17, 18, 19, 20]. However, none of these recognition systems achieves sufficiently fast and accurate perception performance as required by most robotic applications such as object manipulation, autonomous driving, and industrial manufacturing. Major challenges in single-view perception are partial or complete occlusion among object instances, large viewpoint variations of the same object class, and similar appearances shared across different semantic categories. These often occur in densely cluttered scenes, where multiple objects are in close contact and placed over a complex background. The top row of Figure 1.4

shows an example of a cluttered scene from three viewpoints. We can see that a different subset of hand tool objects gets occluded in each view and each object appearance undergoes significant changes during the viewpoint changes. Further, different objects may look similar due to the occlusion and the nature of partial views. These observations reveal the fundamental challenges in single-view perception scenarios which require a perception system to handle large complexity and ambiguity in differentiating objects.

One promising solution to overcome the single-view problems is to fuse predictions from different viewpoints, taking advantage of the fact that multiple scene observations are often available in real perception scenarios such as robotic manipulation and autonomous driving. Recently, various dense SLAM systems such as KinectFusion [21] have emerged for real-time dense 3D reconstruction from consecutive views. They offer fast and reliable camera pose estimation to associate perception results from different frames and establish a geometrically consistent 3D scene model. Such a scene model can represent the foundation for robustly handling occlusions and for carrying out object detection by aggregating object evidence from different viewpoints. The bottom picture in Figure 1.4 illustrates a scene where multiple partial views compensate for each other to generate a complete scene representation. Besides, accurate camera pose estimates provides the foundation for single-view results to vote within a consistent global frame. This enables to resolve the single-view ambiguity and stabilize the long-term perception performance.

Figure 1.4: Figures in the top row show partial observations of the same scene, and the bottom figure demonstrates the reconstructed scene by the SLAM implementation [22]. Objects may be partially or even fully occluded by other objects if observed from a single viewpoint while being fully visible from other viewpoints.

1.2 Outline of Approaches

In this dissertation, we attempt to find solutions to two following critical problems existing in current deep learning architectures.

- Current convolutional architectures employ spatial pooling to achieve scale and shift invariances, but they are still sensitive to out-of-plane rotations. Thus, creating object representations that are robust to changes in viewpoint while capturing local visual details continues to be a challenge. This impedes the progress of learning robust representations for

identifying, localizing and segmenting objects in many robotic applications, where camera viewpoints may change frequently on a moving robot platform.

- Deep models are mainly treated as end-to-end mappings and trained in a pure data-driven manner. Their generalization capabilities are diminished with decreasing data support because the model tends to remember the training data pattern while ignoring the inherent reasoning mechanism. With prior domain knowledge of a complex prediction task, the question remains that whether we can teach a deep model to "think" rather than to "recite".

We present two methodologies to tackle the two problems above, which motivates new types of learning architectures that infer object semantics and geometry from single and multiple views. We first review the related background studies and previous work in Chapter 2. In Chapter 3, we formulate the first methodology called as multi-domain pooling framework which increases the feature robustness to 3D rotations. This supports the object semantics parsing including instance classification and segmentation. Subsequently, we introduce a generalized deep supervision method as the second methodology in Chapter 4. We also demonstrate its applications in learning object geometry properties such as 6-DoF pose and 2D/3D semantic part locations. We extend our single-view approaches to multi-view scenarios in Chapter 5 and conclude our work in Chapter 6

1.2.1 Multi-Domain Pooling

Recent advances in convolutional architectures [9, 23, 24, 25] have achieved success in learning object representations with minor scale and shift invariances. *Spatial Pooling*, which groups local features within spatial neighborhoods, is a key component to achieve those invariance properties. We formulate a probabilistic framework for analyzing the performance of the pooling operation. This framework suggests two directions for reducing the sensitivity to out-of-plane or 3D rotations. First, we make use of additional pooling domains such as color and local gradient pattern, and second we apply multiple scales of filters coupled with different pooling granularities, thereby reducing the sensitivity to spatial deformations. This methodology guides us to design a new type of convolutional architecture using multi-domain pooling, which extracts features for object instance classification and segmentation. We comprehensively evaluate our method on multiple public large-scale benchmarks including UW-RGBD [26], BigBIRD [27], JHUIT-50 [28] and JHUScene-50 [29].

1.2.2 Deep Supervision With Intermediate Concepts

Recent data-driven approaches to scene interpretation predominantly pose inference as an end-to-end black-box mapping, commonly performed by a CNN. However, decades of work on perceptual organization in both human and machine vision suggests that there are often intermediate representations that are intrinsic to an inference task, and which provide essential structure to improve generalization. Therefore, we explore an approach for injecting prior domain structures into neural network training by supervising hidden layers

of a CNN with intermediate concepts that normally are not observed in practice. We formulate a probabilistic framework which formalizes these notions and predicts improved generalization via this deep supervision method. One advantage of this approach is that we are able to train only from synthetic CAD renderings of cluttered scenes, where concept values can be extracted, but apply the results to real images. This lays the foundation for learning generalizable representations to infer 3D shape structures from real images, due to the fact that 3D groundtruth data is scarce in real image datasets but abundant from synthetic CAD rendering pipeline. Our implementation achieves the state-of-the-art performance on 2D/3D keypoint localization on multiple public real image benchmarks including KITTI-3D [30], PASCAL VOC [31], PASCAL3D+ [32] and IKEA [33].

1.2.3 Multi-Class Multi-View Pose Recognition Framework

We present two frameworks for estimating pose across multiple object classes and enhancing the single-view recognition via multiple views. The first framework is based on geometry matching to register object CAD models on regions with homogeneous semantic labels. Semantic labels are inferred using a convolutional architecture with multi-domain pooling which operates on the image time series and which enables efficient propagation of features on the fly. We introduce a probabilistic inference scheme to fuse the semantic prediction results in different views and in turn stabilize the prediction performance. The object registration algorithm is applied on densely reconstructed semantic regions where the scene surface is smoothed and more complete than the one

under single-view.

The second framework aims to learn object 6-DoF pose using convolutional neural network. This learning-based approach exploits the large-scale data to learn pose regression for multiple objects in an unified network architecture. Subsequently, we present a multi-view algorithm to select pose hypothesis, which effectively resolves the pose ambiguity in single-view prediction and is able to continuously improve the pose estimation performance with increasing number of views.

To evaluate these two frameworks, we test our methods on large-scale datasets for object pose estimation: JHUScene-50 [29], ObjectNet-3D [34] and YCB-Video [17]. Our approaches demonstrate improved performance of 6-DoF object pose estimation, compared with current state-of-the-art methods.

1.3 Thesis Statement

An ideal object representation optimizes the trade-off between discrimination and invariance or equivalently bias and variance from a statistical point of view. Creating well-crafted pooling operations provides invariance while maintaining discrimination, and well-chosen model-driven regularization and data fusion improves generalization.

1.4 Contribution

This dissertation is composed of four major contributions detailed in the following sub-sections. The first three of them provide new methodologies

and frameworks for addressing the core challenges in learning object representations for semantics and geometry inference. The remaining one is that we present four object datasets for different object-related vision tasks, including object instance classification, segmentation, pose estimation and sequence-based scene understanding. All these datasets are publicly available at http://cirl.lcsr.jhu.edu/jhu-visual-perception-datasets. In the following, we summarize these contributions and their related publications.

1.4.1 Contribution 1 – Multi-Domain Pooling

We introduce a probabilistic formulation of pooling, associated with an analysis regarding the bias-variance trade-off. This probabilistic perspective of pooling motivates the new designs of convolutional architectures to produce robust representation to 3D rotation by exploiting multiple pooling domains. Further, the multi-domain pooled features can be efficiently computed on time-series RGB-D data, which benefits semantic segmentation in an incremental recognition setting.

1.4.2 Contribution 2 – Deep Supervision With Intermediate Concepts:

We formalize "intermediate concepts" as the latent variables for an inference task. The intermediate concept is coupled with a novel generalized deep supervision scheme, in order to explicitly teach a CNN to recover a series of intermediate goals along the way to the final prediction task. A probabilistic analysis is presented to show its improved generalization capability compared with standard supervision manners such as single-task, multi-task networks

and Deeply Supervised Nets [35].

1.4.3 Contribution 3 – Scalable Learning Architecture for 6-DoF Pose Estimation

We develop a multi-class CNN architecture for accurate pose estimation with three novel features: a) a single, non-branching generic pose representation which induces discriminative features across object categories; b) a method to embed object class labels into the learning process by concatenating a tiled class map with convolutional layers; and c) deep supervision with an object mask is performed so that we can exploit synthetic data to train models that generalize well to real images [30]. With these three innovations, the proposed architecture is readily scalable to large numbers of object categories and works for unseen instances.

1.4.4 Contribution 4 – Multi-View Extension for Semantic Segmentation and Pose Estimation

We present two multi-view frameworks for both object semantic segmentation and pose estimation. Given camera pose estimates from off-the-shelf SLAM systems and single-view perception results, the multi-view fusion algorithms are capable of resolving the single-view ambiguity, aggregating such confidence over time and achieving continuously improved performance in the long term.

1.4.5 Contribution 5 – JHU Perception Datasets

We contribute four RGB-D datasets for object instance classification (JHUIT-50 [28]) and 6-DoF object pose estimation (JHUScene-50 [29], JHUSEQ-25 [36] and LN-66 [37]). In JHUIT-50, we create large viewpoint difference between training and test data in order to evaluate how algorithms generalize across different viewpoints. LN-66 is targeted for recognizing textureless objects in complex assembly structures. JHUScene-50 and JHUSEQ-25 contain densely cluttered scenes where hand tool objects are in close contact and heavy occlusion frequently occurs. These four datasets are critical for the evaluation of our technical contributions presented in this dissertation. We refer readers for more details of each dataset in Appendix 1.5.

1.4.6 Relevant Publications

1. **Chi Li**, Jin Bai, Gregory D. Hager. *A Unified Framework for Multi-View Multi-Class Object Pose Estimation.* European Conference on Computer Vision (ECCV in review), 2018.

2. **Chi Li**, M. Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D. Hager and Manmohan Chandraker. *Deep Supervision with Intermediate Concepts.* IEEE Transactions on Pattern Recognition and Machine Intelligence (TPAMI in review), 2018.

3. **Chi Li**, M. Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D. Hager and Manmohan Chandraker. *Deep Supervision with Shape Concepts for*

Occlusion-Aware 3D Object Parsing. International Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

4. **Chi Li**, Han Xiao, Keisuke Tateno, Federico Tombari, Nassir Navab, Gregory D. Hager. *Incremental Scene Understanding on Dense SLAM.* International Conference on Intelligent Robots and Systems (IROS), 2016

5. **Chi Li**, Jonathan Bohren, Eric Carlson, Gregory D. Hager. *Hierarchical Semantic Parsing for Object Pose Estimation in Densely Cluttered Scenes.* International Conference on Robotics Automation (ICRA), 2016.

6. **Chi Li**, Jonathan Bohren, Gregory D. Hager. *Bridging the Robot Perception Gap With Mid-Level Vision.* Robotics Research, page 5-20, 2017.

7. **Chi Li**, Austin Reiter, Gregory D. Hager. *Beyond Spatial Pooling: Fine-Grained Representation Learning in Multiple Domains.* International Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

The following collaborations and earlier work indirectly contributed to ideas in this dissertation.

1. Chong You, **Chi Li**, Daniel Robinson, Rene Vidal. *A Scalable Exemplar-based Subspace Clustering Algorithm for Class-Imbalanced Data.* European Conference on Computer Vision (ECCV in review), 2018.

1. **Chi Li**, Le Lu, Gregory D. Hager, Jianyu Tang, Hanzi Wang. *Robust Object Tracking in Crowd Dynamic Scenes Using Explicit Stereo Depth.* Asian Conference on Computer Vision (ACCV), 2012.

Figure 1.5: The examples of 50 industrial objects in IT-50 dataset. Each object shown here belongs to a different object instance.

1.5 Appendix: JHU Perception Datasets

In this appendix, we show the details of our four datasets: JHUIT-50 [28], JHUScene-50 [29], JHUSEQ-25 [36] and LN-66 [37].

1.5.1 JHUIT-50

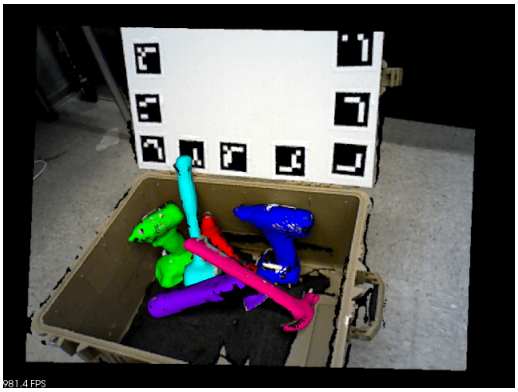
Most RGB-D object datasets only covers a subset of partial views of an object in the entire pose space. Moreover, their training and test data shares substantial overlap in viewpoints. To evaluate the generalization capability across different viewpoints, we present JHUIT-50 dataset which captures a larger range of object poses. In details, JHUIT-50 dataset is captured with an RGB-D camera [1]. It contains 50 industrial objects and hand tools frequently used in

[1] PrimeSense Carmine 1.08 depth sensor is used.

Figure 1.6: A subset of training and testing samples from the object 'drill_flat'. The first three rows in the top block enclosed by a red rectangle show some training samples under viewing angles of 30, 45 and 60 degrees, respectively. The bottom block enclosed by a blue rectangle shows a subset of testing samples captured under random view points.

mechanical operations. Figure 1.5 shows examples of all 50 object instance classes in our IT-50 dataset. To collect the data, we place each object on the center of an electric turn-table with a fixed rotating speed. The camera is fixed on a fixture at different heights and at distance of roughly 1 meter. Training sequences are captured under three fixed viewing angles (30, 45 and 60 degrees) and testing sequences are collected under random view points of the camera. This aims to create large viewpoint difference between training and test data

(a) Hand tool objects



(b) Example groundtruth of object pose

Figure 1.7: Orientations of driving-related objects such as traffic light (1.3a), car (1.3c) and pedestrian (1.3b) are keys for decision making, scene understanding and future prediction.

in order to evaluate how algorithms generalize across different viewpoints. To visualize this, Figure 1.6 demonstrates a subset of the training and testing samples of a drill object. We can see that object appearance changes sharply due to the viewpoint variation. We also provide object masks that segment objects from the background. These masks are automatically generated by ground segmentation and depth filtering, which basically follows the same procedures used for BigBIRD dataset [27].

1.5.2 JHUScene-50

Only a few benchmarks for object pose estimation have been presented in literature. The LINEMOD dataset [5] contains thousands of RGB-D images but only a single pose of a textureless object under almost no occlusion is annotated in each image. The UW-RGBD pose benchmark [26] only provides 1-DoF labeled pose for segmented objects. [38] offers 50 challenging scene frames composed of multiple objects in close contact but no color information

is provided for object models. In this dissertation, we contribute a new scene dataset called JHUScene-50 that is designed to test 6-DoF pose estimation algorithms for generic objects in densely cluttered environments. Compared with previous datasets, JHUScene-50 is more challenging due to heavy occlusion, illumination changes, large viewpoint variation and similar object appearances.

JHUScene-50 contains 50 scenes where each scene has at least three object instances from ten typical hand tools (shown in Fig. 1.7a). For object modeling, we place each of ten hand tools on an electric turntable and capture 900 RGB-D partial views as the training data per object under both fixed and randomly sampled view points[2]. We refer readers to [28] for more details in data collection. We generate a full 3D mesh for each object following similar procedures in [27]. A video sequence of 100 frames is recorded per scene by freely moving a RGB-D camera[3] in one of the five indoor environments including office workspaces, robot manipulation platforms and large containers. Each of the 5 indoor contexts contains ten scenes with multiple object instances densely cluttered in various ways. In addition, we capture a video sequence with 600 frames for each of five indoor environments without any object in it as the training data for the background class.

Each video sequence contains at least 3 hand tool instances that form complex scene clutters, where some objects are in partial and even complete occlusion at some frames. In order to facilitate the annotation for object poses, we place artificial plane markers in the background to compute rigid

[2]Fixed view points are at 30, 45 and 60 degrees above the horizon.
[3]PrimeSense Carmine 1.08 epth sensors.

Figure 1.8: Examples of cluttered scenes in JHUSEQ-25

transformations between first frame and every other frame. We then manually label the 6-DoF poses of all object instances in the first frames and propagate them to the remaining frames. Finally, a post-processing step is conducted to fine-tune propagated poses via Iterative Closest Point (ICP) and remove labeled objects that are not visible in corresponding frames.

In JHUScene-50, there are 22520 labeled object poses in total. Fig. 1.7b shows an example of the labeled object poses. Furthermore, we generate groundtruth of the semantic segmentation of a point cloud by finding nearest vertex of each 3D point among all the annotated poses. If the distance to the nearest neighbor is less than 0.01m, the class label of the corresponding pose is assigned to that point. In turn, all points without any assigned labels belong to the background class.

1.5.3 JHUSEQ-25

JHUSEQ-25 is a large-scale dataset designed specifically for the online sequence-based semantic segmentation, object localization, and 6-DoF pose estimation

in densely cluttered environments. UW RGB-D Scene datasets [39, 40] provide 8 and 14 video sequences per indoor scene with annotations of object locations on the fully 3D reconstructed point cloud. However, they do not provide object pose groundtruth as well as 3D CAD models for furniture objects so that we cannot test our method. Additionally, [29] provides labeled scene frames that are sampled at every one to two seconds, which is not suitable to run dense SLAM systems.

JHUSEQ-25 contains 25 video sequences for 25 different indoor office scenes. Figure 1.8 shows two examples of cluttered scenes. The frame rate is 30fps. Each sequence has 400 frames where each frame is provided with the groundtruth of semantic segmentation, camera and object poses. We manually label the object poses in the reconstructed global scene which shares the same coordinate system as the first frame of each sequence. Then the poses are propagated to the rest frames based on their their camera poses. Object classes in our experiments are 10 hand tools used in [29]. We directly use the object partial views provided by [29] to train the object models [4]. Additionally, we assume that robots know the background prior to the perception. Therefore, we provide a background sequence without any objects from the object dataset, in order to model the background class.

1.5.4 LN-66

Both JHUScene-50 and JHUSEQ-25 contain common hand tools. In the context of industrial manufacturing, we may often deal with small textureless objects

[4]There are 900 partial views per object

or components. To evaluate the pose estimation algorithms in this setting, we create a new LN-66 dataset which contains 66 scenes with various complex configurations of the two "link" and "node" textureless objects shown in Figure 1.1b.

We combine the training and testing sequences (corresponding to fixed and random viewpoints) of "link" and "node" objects in JHUIT-50 [28] as the training data so that each object has 300 training samples. An example testing scene is shown in Figure 1.1c. There are 6 to 10 example point clouds for each static scene from a fixed viewpoint, where each cloud is the average of ten raw RGB-D images. This gives a total of 614 testing examples across all scenes. In our dataset, the background has been removed from each example by RANSAC plane estimation and defining workspace limits in 3D space. Background subtraction can also be done with the semantic segmentation stage if object models are trained along with a background class. Therefore, the points in the remaining point cloud only belong to instances of the "link" or "node" objects. However, robust object detection and pose estimation are still challenging in such scenario due to similar appearances between objects, clutter, occlusion and sensor noise. To quantitatively analyze our method, we manually label the groundtruth object poses for each scene and propagate them to all testing examples. Finally, the groundtruth poses are projected onto 2D to generate the groundtruth for the semantic segmentation at each frame.

Chapter 2

Related Work

In this chapter, we review prior work on object recognition (Section 2.1), object pose estimation (Section 2.2), object geometry learning (Section 2.3) and multi-view perception (Section 2.4). In particular, we discuss the current state-of-the-art approaches and their fundamental limitations in each field.

2.1 Object Recognition

Object recognition is targeted at inferring the category or object identity based on image or other visual input. This also involves distinguishing the object of interest from the background clutter. As such, object recognition can be classified into two categories. The first line of work assumes the groundtruth location of the object is known, and designs methods to identify cropped object images in a large-scale setting (e.g. ImageNet [41]). We review the current state-of-the-art large-scale learning machines in Section 2.1.1. These scalable learning architectures learn robust image representations that benefit many core vision problems including object detection and semantic segmentation,

as detailed in Section 2.1.2.

2.1.1 Large-Scale Learning Machines for Image Classification

AlexNet [9] was the very first successful instance of deep models that leads to the rapid development of deep learning technologies starting from 2012. Traditional neural networks are often shallow in depth (less than two) in order for fast training and inference on CPUs. AlexNet takes the first step to take advantage of highly parallel GPU cores to significantly speed up the training of a 7-layer deep CNN, as shown in Figure 2.1. Additionally, AlexNet introduces two "tricks": Rectified Linear Units (ReLUs) and Dropout, as two fundamental building blocks to improve the training performance and generalization during inference. With these innovations, the original overfitting problem in shallow network is greatly reduced by going deeper. The state-of-the-art performance of AlexNet on ImageNet (in 2012) inspired researchers to revisit the conventional neural network architectures, and paved the way for the booming of various deep models till now.

Two successful successors to AlexNet are GoogleNet [42] and VGG [43]. The core idea of GoogleNet is to learn convolutional filters in multiple local scales at each layer. Two side output layers are inserted at hidden layers as additional supervisions, in order to avoid the vanishing gradient problem. VGG proposes a generic principle of constructing very deep architecture by stacking convolutional layers with small 3x3 filters. Batch Normalization [44] is another critical technique that enables very deep neural networks. At each layer, batch norm operation normalizes the input signal with unit variance

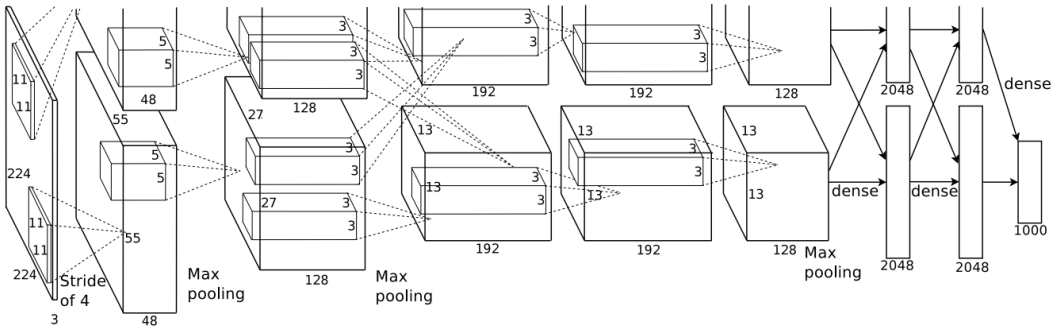


Figure 2.1: Illustration of AlexNet [9] (figure from [9]). It explicitly shows the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers.

and zero mean, which removes "covariate shift [44]" that frequently occurs in the deep network. Stacking of convolutional layers with 3x3 filters and batch normalization layer are two fundamental building blocks for most of state-of-the-art deep learning machines nowadays, including all our network designs presented in this dissertation.

Deep residual network (Res-Net) [45] is one example of pushing the extreme of deep power based on 3x3 filter design and batch normalization. The core innovation is the residual connection between deep layers and shallow layers. This multi-path backpropagation scheme really enforces the learning of an ensemble of neural networks, which further improves the generalization similar to Dropout. Now, Res-Net has reported image classification performance on ImageNet that is competitive to human perception.

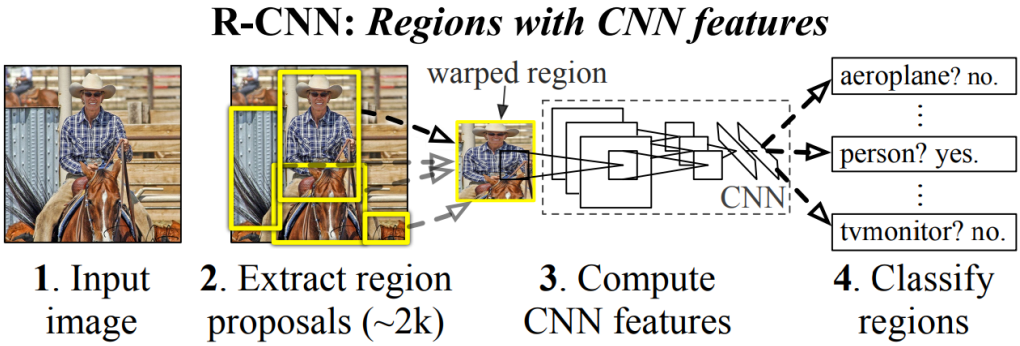


Figure 2.2: Illustration of R-CNN [46] (figure from [46]).

2.1.2 Object Detection and Semantic Segmentation

R-CNN [46] is one early representative of object detection approaches based on large-scale image classification networks. The basic idea is to first extract object proposals (i.e. possible cropped object candidates) and then apply deep CNNs for image classification, as shown in Figure 2.2. Fast R-CNN [47] further improves R-CNN by exploiting a deeper VGG-16 network while speeding up the feature extraction time via RoI pooling. Faster R-CNN deploys a region proposal network to generate object candidates with better quality [48]. Recently, mask R-CNN [49] jointly optimizes detection and instance segmentation and achieves the state-of-the-art performance for both tasks on ImageNet. We can see that the success of most detection systems is achieved based on a robust and discriminative image representation computed from deep CNNs.

Semantic scene segmentation is another well-studied topic. One popular pipeline makes use of Conditional Random Fields (CRFs) to model the pairwise relationships between adjacent local patterns and optimize the overall labeling [50, 51]. However, most of CRF implementations are usually limited

to low-order graphs for the efficiency, which prevent them from modeling patterns over large image areas. Another semantic scene parsing algorithm [52] classifies object proposals generated from a hierarchical scene segmentation tree. A 3-level semantic parsing hierarchy presented in [53] is composed of pixels, supervoxels, and whole instance segments. Unfortunately, these image region hierarchies depend on some bottom-up grouping criteria that are based on strong assumptions such as convex object surfaces. This makes them unable to generalize to broader classes of objects. More importantly, these methods ignore partial object surfaces lying between local regions and global whole-object segments, which makes them sensitive to occlusion. Recently, the state-of-the-art image features learned from Convolutional Neural Networks (CNNs) have been adapted for RGB-D instance segmentation [13] and semantic segmentation [12].

2.2 Object Pose Estimation

Single-view pose estimation can be roughly divided into three main categories: template matching, bottom-up methods and end-to-end learning. We review each of them in the following.

Template Matching. Traditional template-based methods compute object pose by matching image observations to object templates that are sampled from a constrained viewing sphere [5, 16, 20, 54]. One representative is the LINEMOD system [5] which uses gradient templates to match sliding windows to object partial views and initialize Iterative Closest Point (ICP) for pose refinement. This template-based design does not capture fine-grained

visual cues between similar objects and does not scale well to multiple object instances which occlude and/or are in close contact with each other. Furthermore, the precision of LINE-MOD's similarity measure decreases linearly to the increasing percentage of occlusion [55]. Recent approaches apply deep CNNs as end-to-end matching machines to improve the robustness of template matching to partial occlusion and similar appearance across multiple instances [16, 20, 56]. Unfortunately, these methods are not scalable to large-scale problems in general because the inference time grows linearly to the increasing number of objects . Moreover, they generalize poorly to unseen object instances as shown in [20] and suffer from the domain shift from synthetic to real images.

Bottom-Up Approaches. Given object CAD models, the matching of local 3D geometry can be applied to register 3D models into parts of a scene based on coarse-to-fine ICP [57], hough voting [58], RANSAC [4] and heuristic 3D descriptors [59, 60]. More principled approaches use random forest to infer local object coordinates for each image pixel based on hand-crafted features [61, 62, 63] or auto-encoders [64, 19]. Subsequently, energy-based global optimization is used to estimate and refine object poses of multiple instances [61, 63]. However, the local image pattern is ambiguous for objects with similar appearances, which prevents this line of work from being applied to generic objects and unconstrained background clutter.

Learning End-to-End Pose Machines. This class of work deploys deep CNNs to learn an end-to-end mapping from a single RGB or RGB-D image to object pose. [65, 66, 67, 34] directly regress or classify the Euler angles of object

orientations from cropped object images. The main objective of these methods is to recognize object viewpoints from an unconstrained cluttered scenes and generalize to unseen instances of an object category that is trained before. On the other hand, in the context of robotic manipulation, 6-DoF pose is often decoupled into rotation and translation components and each is inferred independently. SSD-6D [18] first predicts discrete rotation bin represented by Euler angle and subsequently estimates 3D position by fitting 2D projections to a detected bounding box. PoseCNN [17] regresses rotation with a loss function that incorporates object geometry into account, and follows bottom-up approaches to vote for 3D location of object center via RANSAC. [68, 69] directly regresses 2D locations of projected bounding box corners and in turn recovers 3D pose from 2D projections via PnP algorithm [70]. Our method formulates a generic and discriminative representation of 6-DoF pose which enables direct prediction of object rotation and translation from either RGB or RGB-D data. Moreover, our approach can be directly applied in unconstrained environment for recognizing viewpoints of unseen instances, in the scale of hundreds of object categories.

2.3 Object Geometry Inference

3D Skeleton Estimation. Many works model 3D shape as a linear combination of shape bases and optimize basis coefficients to fit computed image evidence such as heat maps [71] and object part detections [72]. A prominent recent approach called single image 3D INterpreter Network (3D-INN) [73] is a sophisticated CNN architecture to estimate a 3D skeleton based only on

detected visible 2D joints. However,in contrast to our approach, the training of 3D-INN does not jointly optimize for 2D and 3D keypoint localization. Fitting 3D projection to visible 2D keypoints only can easily lead to incorrect predictions due to the partial view ambiguity and perspective projection, even if they adopt PCA representation to constrain the search space. Also, their stage-wise CNN training pipeline is ad-hoc and not jointly optimized for 2D and 3D keypoint localization. The sampling-based 3D structure fitting [72] integrates the object part detection into objective that jointly models the pose and shape estimation. But it is not robust to complex shapes and inefficient in testing.

3D Reconstruction. A generative inverse graphics model is formulated in [74] for 3D mesh reconstruction by matching mesh proposals to extracted 2D contours. Recently, given a single image, autoencoders have been exploited for 2D image rendering [75], multi-view mesh reconstruction [76] and 3D shape regression under occlusion [77]. The encoder network learns to invert the rendering process to recognize 3D attributes such as object pose. However, methods such as [76, 77] are quantitatively evaluated only on synthetic data and seem to achieve limited generalization to real images. Other works such as [78] formulate an energy-based optimization framework involving appearance, keypoint and normal consistency for dense 3D mesh reconstruction, but require both 2D keypoint and object segmentation annotations on real images for training. Volumetric frameworks using either discriminative [79] or generative [80] modeling infer a 3D shape distribution on voxel grids given image(s) of an object, limited to low-resolutions. However, due to the

highly redundant nature of voxel grid representations, they are limited to low resolutions. Lastly, 3D voxel examplars [81] jointly recognize 3D shape and occlusion patterns by template matching, which is not scalable.

3D Model Retrieval and Alignment. This line of work estimates 3D object structure by retrieving the closest object CAD model and performing alignment, using 2D images [82, 83, 32] and RGB-D data [84, 85]. Unfortunately, a limited number of CAD models can not represent all instances in one object category. Further, the retrieval step is slow for a large CAD dataset and alignment is sensitive to error in estimated pose.

Viewpoint Estimation and 2D Keypoint Detection. "Render for CNN" [65] renders 3D CAD models as additional training data besides real images for object viewpoint estimation. We extend this rendering pipeline to support object keypoint prediction and cluttered scene rendering to learn occlusions from data. Viewpoint prediction is utilized in [86] to boost the performance of 2D landmark localization. Recent work such as DDN [87] optimizes deformation coefficients based on the PCA representation of 2D keypoints to achieve state-of-the-art performance on face and human body. Dense feature matching approaches which exploit top-down object category knowledge [88, 71] are recent successes, but our method yields superior results while being able to transfer knowledge from rich CAD data.

Domain Adaptation. A number of recent works [89, 90, 91] propose to transfer knowledge learned on a labeled "source" dataset, by matching activation statistics of intermediate CNN layers against an unlabeled "target" dataset and employing Maximum Mean Discrepancy (MMD) losses. These

methods have so far been applied to relatively simple datasets, such as MNIST digits and Caltech 101. Our approach explores an orthogonal route to enforce knowledge transfer by employing losses on intermediate tasks. MMD losses can be also leveraged for our method but this exploration lies outside the scope of the present dissertation.

Occlusion Modeling. Most work on occlusion invariant recognition relies on explicit occluder modeling [92, 72]. However, as it is hard to explicitly model object appearance, the variation in occluder appearance is also too broad to be captured effectively by model-driven approaches. This is why recent work has demonstrated gains by learning occlusion patterns from data [93, 81]. Thanks to deep supervision, which enables effective generalization from CAD renderings to real images, we are able to generate and leverage a significantly larger array of occlusion configurations using synthetic data.

Graphics Modeling. Recently, CNNs have been adapted to simulate the graphics rendering by [75] where a compact set of object properties can be mapped to a rendered image describing that object. Kulkarni and et al.[94] takes one step further by adding an encoder module to simulate the inverse graphics process in order to connect the real image and display realistic view. Multi-view CNN [76] shares the same spirit to produce the RGB and depth images given a single image and a pose parameter. However, their objective is to generate high quality synthetic images, which is different from ours. The more recent work [77] applies the CNN to output graphics parameters from the last layer in order for 2D/3D object completion. But it is unknown how it performs on real data and does not conduct the deep supervision as we do.

2.4 Multi-View Perception

One early representative for multi-view object detection [95] tracks feature points across disparate views to learn richer representations of local patterns. Later on, [96, 97] improve monocular object pose estimation via consistency verification over the global geometry estimated by SLAM. These methods are highly limited to objects with distinctive textures and do not scale well with the complexity of object appearances. Lai et al. [39] projects 2D detection scores computed from HOG-based sliding window detectors onto a 3D global model that is built offline. It further corrects the semantic label of each 3D voxel on the global model using Markov Random Field (MRF). Furthermore, [40] designs 3D hierarchical features to directly classify fully reconstructed scenes which contain objects that are well-separated on a flat tabletop or ground plane. A more recent approach [98] achieves better object detection performance by retrieving object candidates from a scale-ambiguous reconstruction map. However, these methods require to use all previous observations whenever the global model is updated. In recent years, several multi-view systems have been developed to enhance 3D model classification [99, 100], 2D object detection [39, 98] and semantic segmentation [36, 101, 57]. But none of them produces estimation of 6-DoF object pose.

Another line of work focuses on jointly estimating camera parameters, scene semantics and 3D geometrical structures. [102] makes use of detected planar objects derived from local feature matching to assist better camera pose estimation during SLAM process and Bao et al. [103] optimizes the semantic labeling, 3D reconstruction and interactions among all scene entities

within a unified graphical model. Unfortunately, the runtime of this system is 20 minutes for each pair of images. In addition, SLAM++ [104] assumes repeatable furniture objects and specific indoor environments for enhanced scene understanding and SLAM performance, which prevent it from generalizing to scenes that contain diverse and unseen object classes. A more recent method [105] formulates a probabilistic framework to fuse pose estimates from different views. However, it requires computation of marginal probability over all subsets of a given number of views, which is computationally prohibitive when the number of views and/or objects is large.

Chapter 3

Multi-Domain Pooling

In this chapter, we introduce the multi-domain multi-scale pooling framework. We first intuitively motivate our approach and give a brief review over the previous work of pooling. Subsequently, we present the probabilistic formulation of pooling which leads to the generalized pooling principles beside spatial pooling. Finally, we show how we apply the multi-domain pooling architectures in instance classification and segmentation.

3.1 Motivation and Overview

The core challenge of object recognition is to create discriminative representations that are robust to appearance variations. Recent advances in convolutional architectures [9, 23, 24, 25] have achieved success in learning object representations with minor scale and shift invariances. *Spatial Pooling*, which groups local features within spatial neighborhoods, is a key component to achieve those invariance properties.

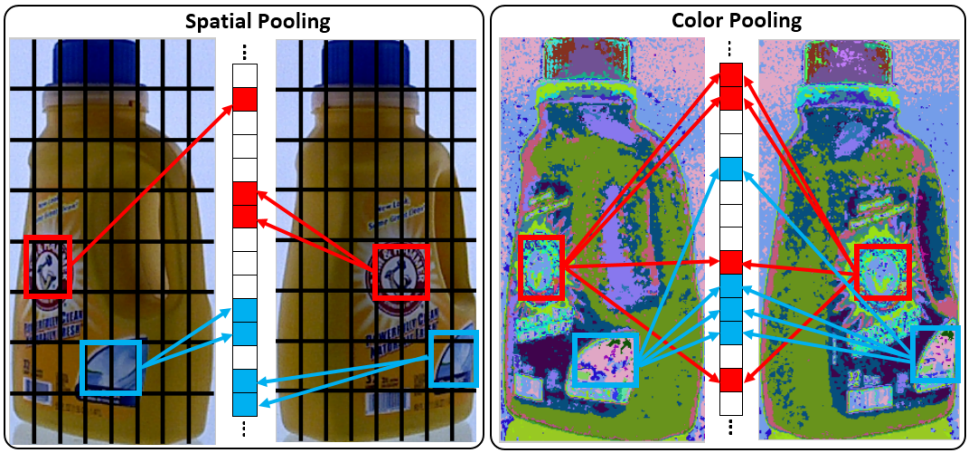


Figure 3.1: A comparison of fine-grained pooling between spatial (X, Y) and color (A, B) (last two channels in CIELAB) domains when an object undergoes a out-of-plane rotation. Fine-grained gridding (8×8) is performed in both domains. Pooling indices in the color domain are shown by different colors in the images. Pooling results for all pixels in two local patterns (enclosed by red and blue rectangles) are shown between pairs of images in each block. Correct feature alignments are made by the color domain, but fail in the spatial domain.

The discrimination and invariance capabilities of the spatially pooled features can be examined with regard to the density of pooling regions which we refer to as *pooling granularity*. The Bag-of-words model, which can be viewed as the extreme case of coarse pooling granularity, can tolerate large variations of object appearances caused by out-of-plane rotations. However, it loses the discriminative power provided by the spatial layout of features [106]. Conversely, fine-grained spatial pooling, which uses small and dense pooling regions (i.e., receptive fields), encodes fine-grained visual cues but is sensitive to spatial rearrangements in different object poses. This is demonstrated on the left block of Figure 3.1, where the same object parts are pooled into different bins under an out-of-plane rotation. One solution is to deploy 'deep' convolutional architectures [9, 107, 108, 109, 110] which hierarchically

pool local responses to boost the discrimination capability of features in the coarse-grained pooling. However, local characteristics are often lost due to the hierarchical pooling. This may not be desirable for the object instance recognition (as opposed to category recognition) where an object should be recognized as exactly the same one that has previously been seen. Recently, [111] integrates part-based modeling [112] into a deep convolutional neural network [9] to create more spatially aligned representations. They achieve state-of-the-art performance in public fine-grained object recognition benchmarks. This implies that robust fine-grained cues can be captured if visual features are better aligned with each other during fine-grained pooling.

Therefore, we analyze the performance of pooling-based convolutional architectures, and propose a simple but effective solution of pooling beyond spatial domain using adaptive scales of filters, to address the feature misalignment problem. Our major innovations are three-fold. First, we formulate a probabilistic framework to mathematically explain how the pooling granularity affects the learned representation in terms of the overall discrimination and invariance. We also argue that fine-grained pooling can be improved with small-scaled filters and invariant pooling domains that are insensitive to object transformations (one example is the color domain shown on the right block of Figure 3.1). Second, based on these ideas, a novel multi-scale and multi-domain pooling algorithm is presented to learn fine-grained representations typical for large-scale object instance recognition task. Small to large scales of filters are coupled with fine to coarse pooling granularities in multiple domains respectively, in order to encode both the localized and global

visual cues. Last, we present a hierarchical semantic segmentation algorithm which fully exploits the multi-domain pooling scheme to efficiently compute features for various segmentation hypotheses. Driven by multi-domain pooling, this segmentation algorithm shows significant improvement over the state-of-the-art methods especially in densely cluttered scenes.

We comprehensively evaluate our methods on four public RGB-D benchmarks [26, 27, 28, 29] for single-view instance classification and segmentation.

3.1.1 Related Work

Invariant representation learning has been studied in the past with empirical validations [113, 114, 115, 116] and theoretical analyses [117, 118]. Spatial pooling is found to be critical to gain the shift invariance in both feature coding pipelines [106, 119, 120, 121] and deep convolutional neural networks [9, 109, 110, 122]. Recently, an unsupervised feature learning theory [118] proposed an invariant signature by characterizing the distribution of template responses within certain transformation groups. This idea is shared in the design of the TIRBM [123], where minor 2D affine transformations are modeled during training. Similarly, data augmentation, a trick commonly used in deep CNNs [122, 9], is functionally equivalent to this strategy. However, in this category of work, only invariance to 2D affine transformations at most can be guaranteed for general object classes and only a subset of transformations can be modeled in practice. To resolve above issues, our method directly exploit invariant pooling domains to learn feature.

Pooling in input feature space [124, 125, 126] can smooth the representation for better invariance, but this tends to lose discrimination capabilities. Thus, spatial layouts [124, 125] or supervised labels [127] are employed to create discriminative features. Additionally, learning optimal spatial pooling configurations in multiple pooling scales has been attempted by supervised [128, 129, 130] and unsupervised [131, 132] techniques as well as segmentation priors [133]. This series of work uses fixed filter scales in the spatial pooling domain while our method couples the adaptive filter scales with pooling granularities and deploys additional pooling domains to overcome feature misalignments.

Various rotationally invariant 3D feature descriptors [134, 59, 135, 136] were proposed for 3D object recognition, but these have been out-performed by multi-cue kernel descriptors [137, 25] and hierarchical convolutional architectures [107, 138, 108] in large-scale settings [26, 27]. The state-of-the-art method [107] mainly uses high-level features, coarse-grained spatial pooling, and contrast normalization to alleviate large intra-class variance caused by 3D rotations. However, spatial pooling still dominates the feature learning in those approaches, which makes learned representations only invariant to limited views of an object. In this study, we demonstrate that pooling simple local features in invariant domains can significantly boost the recognition performance for the object instance recognition.

3.2 Methodology

An overview of the general pooling process in a convolutional architecture is shown in Figure 3.2. Filter responses associated with each pooling state are activated by feature filters convolved over visual signals. In the case of spatial pooling, pooling states are pixels in normalized image coordinates. A pooling operator extracts some statistics over filter responses within neighborhoods of pooling states. Few theoretical investigations have been presented in the literature to explain why pooling is critical in creating invariant representations. One pooling theory was proposed by Boureau [139] in the context of hard-assignment coding. It assumes that filter responses in a pooling region have identical and independent Bernoulli distributions given an object class. These conditions restrict the theory from generalizing to more complex scenarios. In this section, we develop a novel probabilistic view for pooling to resolve the aforementioned issues, which in turn motivates the proposed generalized pooling principles in Section 3.2.4.

3.2.1 Interpretation of Discrimination and Invariance

Consider a pooling domain $S = \{s_1, \cdots, s_N\}$ where pooling state s_j with $1 \leq j \leq N$ is a coordinate over which pooling takes place. For example, in the case of RGB-D data, S can be a set of spatial coordinates or color values, corresponding to spatial and color domains.

We now introduce a set of K filters $D = \{d_1, d_2, \cdots, d_K\}$. In the context of feature coding, these filters are codewords learned by dictionary learning techniques. Note that filters are not necessarily defined over the pooling domain

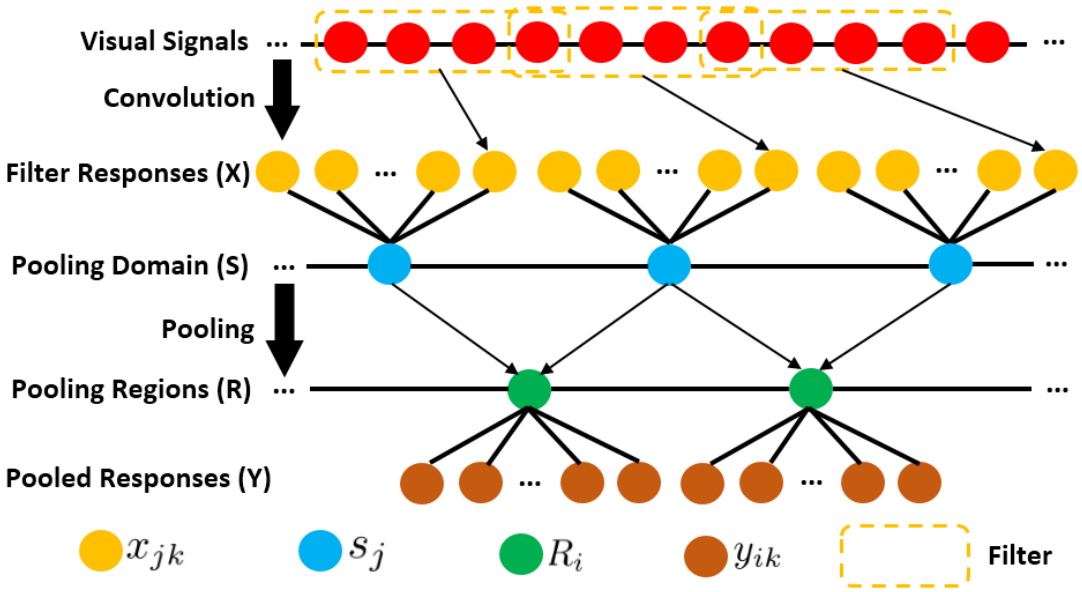


Figure 3.2: Demonstration of a general pooling process and related notations used in this section. At the top layer, we only show the convolution of one filter with one single scale. [Best viewed in color]

(e.g., we could use the color domain to pool responses from spatial filters). Next, we define $X = (x_{11}, \cdots, x_{jk}, \cdots, x_{NK})$ as non-pooled representation for a data sample, where each $x_{jk} = (s_j, d_k)$ captures the activation strength of d_k at s_j (second row of Figure 3.2). Each visual signal that occupies s_j contributes its K filter responses to the part of X associated with s_j. If two or more signals fall into the same s_j, we could compute the final response for each x_{jk} using any statistics (maximum value for example). Considering a random sampling of images generated by applying some transformation function $\mathcal{T}$ for object o_p, let $X^p = (x^p_{11}, \cdots, x^p_{jk}, \cdots, x^p_{NK})$ denotes the random vector of the filter responses with the distribution $P(X^p) = P(X|o_p)$. The $P(X^p)$ characterizes the distribution of the set of filter responses $G = \{X^p_i\}$ where X^p_i is a sample of X^p generated by $\mathcal{T}$.

We measure the variability of X^p with an invariance score J^p. Specifically, J^p is defined as the average Euclidean distance[1] between all samples in G:

$$\begin{aligned} J^p &= \frac{1}{t^2}\sum_{i=1}^{t}\sum_{j=1}^{t}\|X_i^p - X_j^p\|_2^2 = E(||X^p - \widetilde{X}^p||_2^2) \\ &= \sum_{j=1}^{N}\sum_{k=1}^{K} 2\mathrm{Var}(x_{jk}^p) \end{aligned} \tag{3.1}$$

where $X_i^p, X_j^p \in G$. We use X^p and $\widetilde{X}^p$ as random variables for $\{X_i^p\}$ and $\{X_j^p\}$ respectively, which share the same distribution $P(X^p)$. As we can see, the invariance score J^p is actually the sum of variances of all dimensions in X^p. It measures how concentrated the representation is under the transformation $\mathcal{T}$. The smaller J^p, the better the stability of the descriptor.

Next, we formulate a distance metric $\mathcal{D}(X^p, X^q)$ between X^p and X^q given two object classes o_p and o_q as follows:

$$\mathcal{D}(X^p, X^q) = \frac{1}{2}\frac{\|\Delta E\|_2^2}{J^p + J^q} \tag{3.2}$$

where $\Delta E = E(X^p) - E(X^q)$. We could interpret the numerator and denominator in $\mathcal{D}(X^p, X^q)$ as the measurements of the discrimination and invariance properties of non-pooled representation X, respectively. In fact, $\mathcal{D}(X^p, X^q)$ can be derived as the lower bound of the Bhattacharyya distance metric

[1]This corresponds to the distance metric in linear SVM which is used in this study.

$\mathcal{D}_B(X^p, X^q)$ given that $P(X^p)$ and $P(X^q)$ follow multivariate normal distributions with covariances Σ_p and Σ_q. This can be shown as follows:

$$\begin{aligned}
\mathcal{D}_B(X^p, X^q) &= \frac{1}{8}\Delta E^\top \bar{\Sigma}^{-1}\Delta E + \frac{1}{2}\ln\frac{|\Sigma|}{\sqrt{|\Sigma_p||\Sigma_q|}} \\
&\geq \frac{1}{8}\Delta E^\top (U\bar{\Lambda}^{-1}U^\top)\Delta E \\
&\geq \frac{1}{8}\frac{\|U^\top \Delta E\|_2^2}{tr(\bar{\Lambda})} \\
&= \frac{1}{2}\frac{\|\Delta E\|_2^2}{J^p + J^q} \\
&= \mathcal{D}(X^p, X^q)
\end{aligned} \tag{3.3}$$

where $\bar{\Sigma} = \frac{\Sigma_p + \Sigma_q}{2}$ with eigen-decomposition $\bar{\Sigma} = U\bar{\Lambda}U^\top$. The second step is obtained by the Cauchy-Schwarz inequality and the third step is derived according to the mediant inequality [2]. The final step follows by $\|Ux\| = \|x\|$ if U is unitary and the $tr(\bar{\Lambda}) = tr(\bar{\Sigma}) = \frac{1}{4}(J_p + J_q)$. **Note that random variables are allowed to be dependent on each other in this derivation.** From the perspective of the lower bound of $\mathcal{D}_B(X^p, X^q)$, $\mathcal{D}(X^p, X^q)$ characterizes the most ambiguous region between two feature distributions. Notice that $\mathcal{D}(X^p, X^q)$ shares a similar form with the objective in linear discriminant analysis (LDA) and distribution separability in [139] using a signal-to-noise ratio.

[2] $\frac{b}{a} + \frac{d}{c} \geq \frac{b+d}{a+c}$ if $a, b, c, d \geq 0$

3.2.2 Variance Reduction Via Pooling

In this section, we show that pooling filter responses within regions in S reduces the variance of the non-pooled representation X^p. Let $\mathcal{R} = \{R_1, \cdots, R_M\}$ be a partition of S (i.e., a set of non-overlapping pooling regions) and assume max pooling is used [3]. In turn, we define a new random variable $y_{ik} = \max_{s_j \in R_i} x_{jk}$ that represents the pooled filter response in pooling region R_i. Analogous to X^p, we then define the random vector $Y^p_{\mathcal{R}} = (y^p_{11}, y^p_{12}, \cdots, y^p_{MK})$. $J^p_{\mathcal{R}}$ is the invariance score of the pooled representation $Y^p_{\mathcal{R}}$. In turn, we can prove the following result based on Theorem 1 that $\text{Var}(\max_i X_i) \leq \sum_i \text{Var}(X_i)$ in Appendix 3.7:

$$J^p_{\mathcal{R}} = \sum_{k=1}^{K} \sum_{i=1}^{M} 2\text{Var}(y^p_{ik}) \leq \sum_{k=1}^{K} \sum_{j=1}^{N} 2\text{Var}(x^p_{jk}) = J^p \tag{3.4}$$

In short, max pooled feature $Y^p_{\mathcal{R}}$ has lower variance than non-pooled feature X^p, which means $Y^p_{\mathcal{R}}$ is less sensitive to transformations $\mathcal{T}$ than X^p. The same can be shown for average pooling [4] because $\text{Var}(\frac{1}{N}\sum_i X_i) \leq \sum_i \text{Var}(X_i)$. Furthermore, J^p is a very loose upper bound for $J^p_{\mathcal{R}}$ in Equation 3.4. The equality is achieved in the asymptotic regime when one random variable is always greater than remaining ones with zero variance. Therefore, $J^p_{\mathcal{R}}$ is much smaller than J^p in practice.

Furthermore, in the case of intersecting pooling regions $\widehat{\mathcal{R}} = \{\widehat{R}_1, \cdots, \widehat{R}_M\}$, we can find a non-overlapping set $\mathcal{R} = \{R_1, \cdots, R_M\}$ subject to $\cup R_i = \cup \widehat{R}_i$

[3] We choose max pooling operator [140] for our main analysis because many studies [139, 121] show its better performance over average pooling.

[4] It is equivalent to sum pooling in the context of Equation 3.5

and $R_i \subseteq \widehat{R}_i$. Then we can get $J^p_{\widehat{\mathcal{R}}} \leq J^p_{\mathcal{R}}$ because each $\widehat{R}_i$ further pools the result of R_i so that the invariance score decreases according to Equation 3.4. Thus, the overlapping pooling scheme achieves even lower variance than the non-overlapping case, though it tends to decrease $\|\Delta E_{\widehat{\mathcal{R}}}\|_2^2 = \|E(Y^p_{\widehat{\mathcal{R}}}) - E(Y^q_{\widehat{\mathcal{R}}})\|_2^2$ since each pooling region is more likely to acquire high activation responses when it is enlarged. For simplicity, we continue to assume pooling regions are a partition in the following discussion.

Analogous to Equation 3.3, we can also write the distance between $Y^p_{\mathcal{R}}$ and $Y^q_{\mathcal{R}}$ for object classes o_p and o_q as follows:

$$\mathcal{D}(Y^p_{\mathcal{R}}, Y^q_{\mathcal{R}}; \mathcal{R}) = \frac{1}{2} \frac{\|\Delta E_{\mathcal{R}}\|_2^2}{J^p_{\mathcal{R}} + J^q_{\mathcal{R}}} \tag{3.5}$$

where $\Delta E_{\mathcal{R}} = E(Y^p_{\mathcal{R}}) - E(Y^q_{\mathcal{R}})$. It is clear that greater discrimination $\|\Delta E_R\|_2^2$ and lower variance $J^p_{\mathcal{R}} + J^q_{\mathcal{R}}$ lead to better separability and in turn easier classification.

3.2.3 Discussion and Conclusion

The above probabilistic framework for pooling yields three major conclusions:

1. As pooling granularity changes from fine to coarse levels, pooled features have better invariance (smaller $J^p_{\mathcal{R}}$) but less discrimination (smaller $\|\Delta E_{\mathcal{R}}\|_2^2$).

2. Small-scale filters achieve better invariance than the large-scale ones in fine-grained pooling.

3. Pooling domains that are insensitive to transformations obtain better invariance in fine-grained pooling.

The first point follows from to Equation 3.4. $J^p_{\mathcal{R}}$ is monotonically decreasing (i.e., invariance of $Y^p_{\mathcal{R}}$ is increasing) with growing size of pooling regions. This can be shown by replacing the left and right sides in Equation 3.4 with variances of pooled features from small and large pooling regions, respectively. On the other hand, the discrimination term $\|\Delta E_{\mathcal{R}}\|_2^2 = \sum_{k=1}^{K}\sum_{j=1}^{M}|E(y^p_{jk}) - E(y^q_{jk})|^2$ tends to decrease due to smaller M at a coarse pooling granularity, especially when y_{jk} is bounded in most of the feature encoding algorithms. One good tradeoff between invariance $J^p_{\mathcal{R}} + J^q_{\mathcal{R}}$ and discrimination $\|\Delta E_{\mathcal{R}}\|_2^2$ to get large $\mathcal{D}(Y^p_{\mathcal{R}}, Y^q_{\mathcal{R}}; \mathcal{R})$ is made by 'deep' representations [9, 107, 138, 108, 23, 109, 110], which augments discrimination capabilities in coarse-grained pooling with highly class-specific filters. In this work, we pursue a good tradeoff along the other direction in which the feature invariance is enhanced in fine-grained pooling.

Next, we jointly analyze the last two points by looking more closely at $\mathrm{Var}(x^p_{jk})$. In the context of fine-grained pooling where the number of pooling regions M is large, the invariance score $J^p_{\mathcal{R}}$ significantly drops if the variance of filter responses at each pooling state $\mathrm{Var}(x^p_{jk})$ is reduced whereas the discrimination term $\|\Delta E_R\|_2^2$ is dominated by M and remains roughly the same. Therefore, we explore two ways to reduce $\mathrm{Var}(x^p_{jk})$ for better separability $D(Y^p_{\mathcal{R}}, Y^q_{\mathcal{R}})$ in fine-grained pooling. Specifically, we observe that $P(x^p_{jk})$ can be

decomposed into the following two forms:

$$P(x_{jk}^{p}) = P(d_k|s_j, o_p)P(s_j|o_p) \tag{3.6}$$

$$P(x_{jk}^{p}) = P(s_j|d_k, o_p)P(d_k|o_p) \tag{3.7}$$

As a result, $\mathrm{Var}(x_{jk}^{p})$ is positively proportional to $\mathrm{Var}(d_k|s_j, o_p)$ or $\mathrm{Var}(s_j|d_k, o_p)$ [5]. Then we could make $\mathrm{Var}(x_{jk}^{p})$ smaller by decreasing either $\mathrm{Var}(d_k|s_j, o_p)$ or $\mathrm{Var}(s_j|d_k, o_p)$. First, reducing $\mathrm{Var}(d_k|s_j, o_p)$ can be interpreted as choosing filters that have smaller variance across the pooling domain S. Given a fixed filter learning method, smaller $\mathrm{Var}(d_k|s_j, o_p)$ is achieved via small-scale filters rather than large-scale ones because the value changes of local regions are less than large areas in convolution. However, large-scale filters are prone to create better discrimination, which is more favored in coarse-grained pooling. Second, reducing $\mathrm{Var}(s_j|d_k, o_p)$ is equivalent to constructing a pooling domain where appearance features have better alignments at each s_j. In other words, a more robust pooling domain with respect to transformations leads to smaller variance of filter responses at each pooling state s_j. Considering 3D transformations, spatial layouts of the transformed object samples change sharply while color configurations are typically aligned across different poses [6]. The possible color misalignment is caused by different lighting conditions, which can be largely alleviated by a good choice of color space and the pooling process. This fact motivates us to exploit the color domain as an example of

[5] This is proven by Theorem 3 in the supplementary material

[6] Photometric variation of object appearances are much smoother in general.

an invariant domain in this study .

Although the spirit of discrimination-invariance tradeoff is already revealed by some kernel learning techniques [141], our framework associates it with pooling operator in the context of the convolutional architecture. As far as we know, we are the first to present this novel view and explore the way to make a good tradeoff. All the three conclusions derived in this section are empirically validated in Section 3.2.5.

3.2.4 Generalized Pooling Principle

Consider a set of pooling domains $S = \{S^t\}$ and a point cloud $P = \{p_i\}$ with its corresponding local feature $X = \{x_i\}$. Here, the local feature X can be arbitrary fixed dimensional feature vectors from CNNs or other hand-crafted features. Our objective is to determine a method to pool X in S. We define a pooling pair $< c_i^t, x_i >$ for a 3D point p_i, where pooling indicator $c_i^t \in S^t$ is used to direct the local response x_i to a specific pooling region in S^t. Essentially, c_i^t is a feature representation for p_i in S^t (e.g. $c_i^t \in S^t$). For example, if S^t denotes the SIFT feature space, c_i^t is a SIFT feature descriptor of p_i. Next, we consider a set of region seeds $W^t = \{w_j^t \mid w_j^t \in S^t \wedge 1 \leq t \leq M\}$ as the representation of M pooling regions $R^t = \{r_1^t, \cdots, r_M^t\}$ in S^t. The ith pooling region r_i^t is defined as:

$$r_i^t = \{w \mid (w \in S^t) \wedge (i = \arg_j \min \|w - w_j^t\|)\} \tag{3.8}$$

Instead of explicitly specifying the sizes and locations of each pooling region in R^t, the configuration of pooling regions is implicitly described by the online

nearest neighbor search among region seed set W^t. That is, the local feature x_i is pooled in region R_j^t if and only if w_j^t is the nearest neighbor of c_i^t among W^t.

In order to capture richer visual characteristics, we can deploy multiple seed sets $\{W_k^t\}$ in one pooling domain to build a set of pooling regions $\{R_k^t\}$ with different pooling granularities [28]. We note that spatial and color pooling are special cases of the above generic pooling scheme. In color pooling, for example, c_i^t is the color value of p_i and W^t is the set of centers of gridded cells in color space. The pyramid structure over the spatial or color domain can also be constructed by setting $\{W_k^t\}$ as the centers of pooling cells at each level.

Next, the sum-pooling operator is applied to sum over all local responses that go into the same pooling region. We use it because of its better performance than the max-pooling operator. Thus, the pooled representation $Y(R_k^t)$ for R_k^t is a concatenation of $L2$-normalized pooled features from all M pooling regions $R_k^t = \{r_{1,k}^t, \cdots, r_{M,k}^t\}$:

$$Y(R_k^t) = [Y(r_{1,k}^t), \cdots, Y(r_{M,k}^t)] \tag{3.9}$$

where the ith dimension $Y(r_{i,k}^t)$ is computed as follows:

$$Y(r_{i,k}^t) = \frac{\widehat{Y}(r_{i,k}^t)}{\|\widehat{Y}(r_{i,k}^t)\|_2}, \;\; s.t. \; \widehat{Y}(r_{i,k}^t) = \sum_{c_i^t \in r_{i,k}^t} x_i \tag{3.10}$$

Suppose that we have T pooling domains and $\{K_1, \cdots, K_T\}$ are the numbers of pooling seeds used for all these T domains. We concatenate the pooled

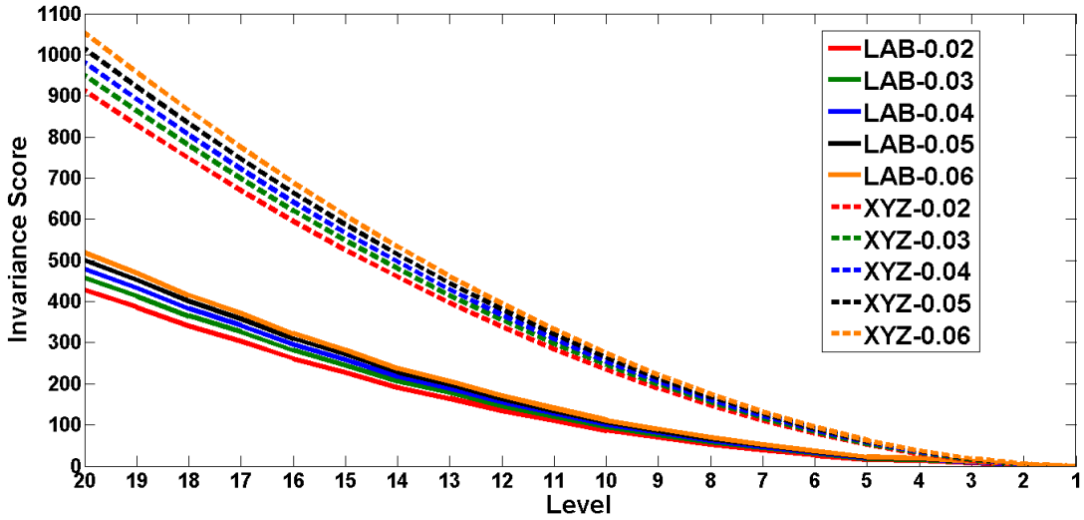


Figure 3.3: Comparison of the variances in different filter scales, pooling granularities and domains. The legend name 'domain-radius' indicates the pooling domain and the radius of CSHOT features respectively. [best viewed in color]

representations in all T pooling domains as the final data representation:

$$Y = [Y(R_1^1, \cdots, R_{K_1}^1, \cdots, R_1^T, \cdots, R_{K_T}^T)] \tag{3.11}$$

3.2.5 Empirical Validation

We first conduct an experiment to verify the three conclusions derived from the probabilistic framework in Section 3.2.3. The experimental results obtained in this section are commonly observed in almost all objects in UW-RGBD [26], BigBIRD [27] and JHUIT-50 [28]. For simplicity, we choose the object 'mixed_berry'[7] from BigBIRD [27] as the representative for analysis. The variance in object representation is rooted from different object poses under 3D transformations. A detailed description about the object data can

[7]It is short for 'eating_right_for_healthy_living_mixed_berry'.

be found in Section 3.3.3. We compute the local response X by extracting the CSHOT descriptor [59]. Concretely, CSHOT features with radii ranging from 0.02m to 0.06m are extracted on every 3D points from an object point cloud and pooled from level-1 to level-20 separately in both the XYZ and LAB domains. Figure 3.3 shows the empirical invariance scores of Equation 3.1 across different levels and domains. Three major observations follow: (1) The invariance of the representation generated by all scales of filters in either domain increases via pooling [8] and maximal invariance is achieved by pooling in the entire domain (i.e., bag-of-words model). (2) Large-scale filters retain greater variance in all levels and both domains than small-scale filters. (3) The color domain exhibits much less variance in the learned representation than the spatial domain in all pooling granularities. These three observations empirically verify the three major points concluded in Section 3.2.3. This further supports the proposed multi-domain pooling algorithm for instance segmentation in Section 3.3.1.

3.3 Instance Recognition

For instance recognition, a vision system should classify exactly the same instance as it has seen before. Because different instances in one object category may share similar texture or shape, a good representation for instance classification task should preserve the visual details while being robust to 3D object transformation. This motivates us to take advantage of multi-domain pooling principles presented in Section 3.2.4.

[8] Smaller invariance score indicates better invariance.

3.3.1 Algorithm and Implementation

The three theoretical views shown in Section 3.2.3 directly lead to the design of the multi-scale and multi-domain pooling algorithm presented in this section. Prior to going into the details of the proposed method, we first briefly explain the local features we use. First, we use CSHOT [59], a rotationally invariant 3D feature, as the local descriptor of a 3D point p_i in a point cloud $P = \{p_1, \cdots, p_n\}$. We modify the original CSHOT descriptor by decoupling the color and depth components. Next, by using via hierarchical K-means, two separate dictionaries D_c and D_d with K filters (e.g. codewords) are learned for both $L2$-normalized color and depth components (denoted as z_c and z_d) in CSHOT. We do so by randomly sampling raw CSHOT features across different object classes as the training data and run hierarchical K-means. In turn, given a test point p_i, we compute its responses with respect to both CSHOT dictionaries by transforming each of its CSHOT component z (e.g. z_c or z_d) into a response vector $\mu = \{\mu_1, \cdots, \mu_j, \cdots, \mu_K\}$ by the soft-assignment encoder [142]:

$$\begin{aligned} \mu_j &= \frac{\exp\left(\beta \hat{d}(z, d_j)\right)}{\sum_{k=1}^{n} \exp\left(\beta \hat{d}(z, d_k)\right)} \\ s.t.\ \hat{d}(z, d_k) &= \begin{cases} d(z, d_k) & : d_k \in \mathcal{N}_k(z) \\ +\infty & : d_k \notin \mathcal{N}_k(z) \end{cases} \end{aligned} \tag{3.12}$$

where $\mathcal{N}_k(z)$ denotes the k-nearest neighbors of a raw CSHOT component z defined by the Euclidean distance $d(z, d_k)$ between z and the kth codeword d_k. β is a smoothing parameter with negative value. Finally, the local response

x_i for p_i is constructed by concatenating the responses μ_c and μ_d for z_c and z_d: $x_i = [\mu_c, \mu_d]$. We keep the feature extraction simple in order to isolate the contributions in our proposed pooling algorithm.

Unlike spatial pyramid pooling, where filter responses with fixed scale go into different pooling levels, the second point in Section 3.2.3 inspires us to pool responses from small-scale filters in fine-grained levels while large-scale filter responses are pooled in coarse-grained levels. In our implementation, we adjust the scales of filters (i.e., codewords) by altering the 3D radius of CSHOT feature.

Following the generalized pooling principles in Section 3.2.4, we employ the three additional pooling domains: color (LAB space), SIFT [143] (gradient) and FPFH [60] (3D geometry), which are robust to 3D rotation. Both SIFT and FPFH domains are also invariant to illumination change. Therefore, each CSHOT filter response goes into a pooling region based on the color, SIFT and FPFH descriptors of the RGB-D image pixel associated with it and the sum pooling is applied for all responses within the same pooling region. Note that spatial domain may not be abandoned because spatially aligned features under slight change of view points could still benefit the recognition (shown in Section 3.3.2).

In summary, the proposed method (shown in Figure 3.4) is evolved from the common coding-pooling pipeline [106, 121, 24], but it conducts an adaptive pooling scheme on convolutional filter responses in multiple scales and additional pooling domains. Pooled features from fine to coarse pooling levels across different domains are concatenated together to generate the final

representation and a linear SVM is used for the classification.

Implementation Details We perform experiments on three RGB-D datasets: UW-RGBD[26], BigBIRD[27] and our own JHUIT-50 dataset [28]. CSHOT features are extracted densely over each point in the point cloud that is generated from color and depth images. We alter the radius of the CSHOT feature to adjust the scale of the filters. Depth and color components in the raw CSHOT feature are decoupled into two feature vectors. Dictionaries with 200 codewords are learned by hierarchical K-means for each component. Note that the dictionary size is fixed across CSHOT filters with different radii. Finally, a soft-assignment encoder [144, 145] is used to generate feature codes of both components which are further concatenated as the local feature code. We choose the number of nearest neighbors K as $K = 20$ and the smoothing factor β as $\beta = -4.0$ in soft encoding (Equation 3.12). All parameters are selected by cross-validation on a subset of the UW-RGBD dataset [9]. Feature codes within the same pooling region are further normalized using the L2-norm.

We choose the CIELAB color space as the color pooling domain since we found that it achieves better performance than both RGB and HSV color spaces. The spatial domain is constructed in 3D space (XYZ). Each channel in the spatial and color domains is normalized to $[0, 1]$ to gain scale invariance. The feature codes are pooled inside the cells of the pyramid with multiple levels. Each level is constructed with a different granularity by gridding in a particular domain. Specifically, level-k in either the spatial (XYZ) or color (LAB) domain is constructed by $k \times k \times k$ grids. Pooled features across

[9] First 30 object instances.

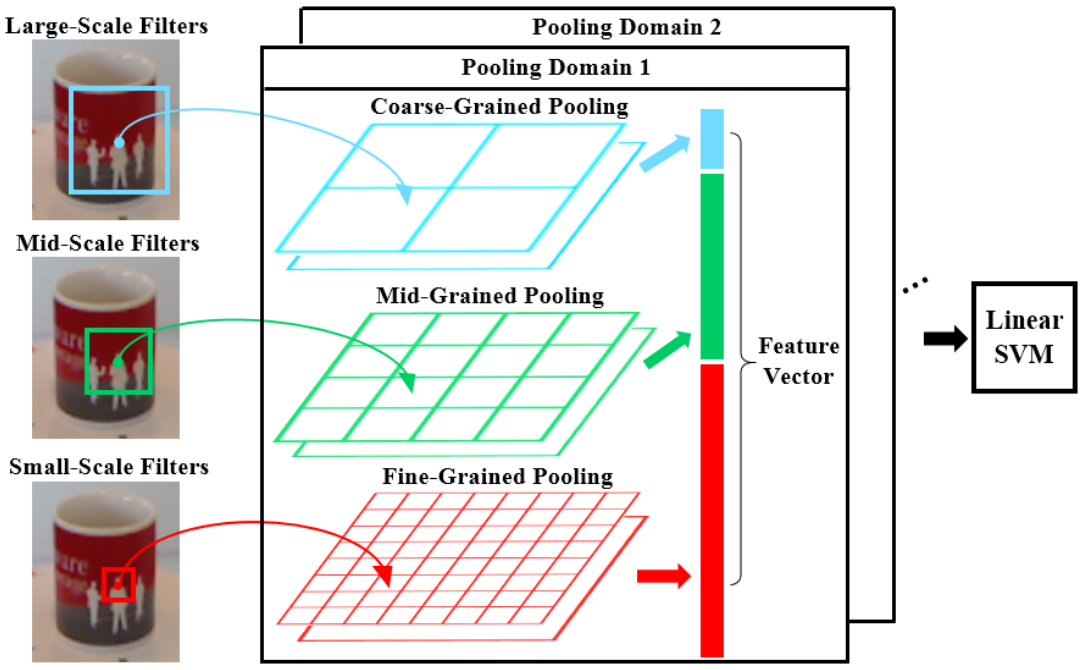


Figure 3.4: Overview of multi-scale and multi-domain pooling architecture.

different levels and domains are concatenated as the final representation.

For SIFT and FPFH domains, we randomly sampled SIFT or FPFH features from UW-RGBD are clustered via hierarchical K-means to learn 400 region seeds for each of the FPFH or SIFT pooling domains, respectively. That is, we have only one pooling resolution for SIFT and FPFH domains. For SIFT pooling, we run multiple SIFT keypoint detectors with $\sigma \in (0.7, 1.6)$ and use all keypoints by setting no threshold response on edges and corners[10]. For FPFH pooling, we compute FPFH descriptors for each 3D point with radius 0.02m and normalize them with L2-norm. Keypoints with valid CSHOT codes are pooled by finding $K = 20$ nearest neighbors among 400 region seeds based on their SIFT and FPFH descriptors, respectively.

[10]We use the non-free SIFT implementations in OpenCV.

Algorithm	Accuracy	Algorithm	Accuracy
XYZ-S-1	75.1	LAB-S-1	75.1
XYZ-S-2	84.3	LAB-S-2	87.8
XYZ-S-3	86.1	LAB-S-3	88.3
XYZ-S-4	85.7	LAB-S-4	89.2
XYZ-S-5	85.5	LAB-S-5	89.6
XYZ-M-5	87.9	LAB-M-5	91.9
ALL-S-5	93.3	ALL-M-5	94.1

Table 3.1: Testing accuracies (%) of different number of stacked levels and filter scales in spatial (XYZ) and color (LAB) domains. "S" indicates single scale filter. "M" indicates multi-scale filter. "All" means both XYZ and LAB domains are used.

3.3.2 UW-RGBD

We perform ablative studies and comparative experiment on the UW-RGBD object benchmark [26]. The UW-RGBD dataset contains 300 textured and texture-less object instance classes. All models are trained on training split provided from UW-RGBD and test under the leave-sequence-out setting.

We first examine how pooling granularity and multi-scale filter scheme affects the instance classification performance between color and spatial domains. Table 3.1 reports the accuracies achieved by different variants of the proposed method. These variants vary in different numbers of stacked levels and filter scales in XYZ and LAB pooling domains. The algorithm name is formatted as 'domain-type-level'. More specifically, 'domain' indicates the pooling domain from LAB, XYZ or both, 'type' includes 'S' and 'M' referring to single and multiple scales of filters, and 'level' specifies the number of stacked levels used in the pyramid from level-1. For type 'S', we use the CSHOT feature with radius 0.03 across all experiments. In type 'M', feature responses from five scales of CSHOT filters from 0.02m to 0.06m with interval 0.01m

are pooled within levels from 5 to 1 respectively. Accuracies in level-1 are the same between XYZ and LAB because the bag-of-words modeling results in the same pooled features regardless of the domain. Beyond level-1, the color domain consistently achieves higher accuracies than the spatial domain. Also, when pooling is performed over fine-grained levels, color pooling is able to continuously boost the recognition rates while spatial pooling fails to do so. This observation substantiates that the better invariance achieved by the color domain (shown in Figure 3.3) helps to utilize the discrimination power in fine-grained levels.

We also observe that the XYZ domain performs worse than the LAB domain and the combination of both domains achieves the best performance. This is because the view point changes in this experiment design ($15 \sim 20$ degrees) do not significantly disrupt the spatial layout for some typical objects with nearly homogeneous appearances, like a ball. Thus, correct feature alignments can be captured by spatial pooling to benefit the overall recognition. Lastly, when we couple the fine-to-coarse pooling granularities with small-to-large filter scales, this multiple-scale filter (M) is consistently superior to single single- scale filter (S). This empirically supports the second conclusion in Section 3.2.3. Finally, the multi-scale and multi-domain pooling scheme ('All-M-5') achieves the best result at 94.1%. Therefore, pooled features from spatial and color domains can complement to each other and improve the performance obtained by each individual domain.

Next, we inspect more additional pooling domains including gradient (SIFT) and 3D geometry (FPFH). Table 3.2 reports the recognition accuracies of

Algorithm	Accuracy	Algorithm	Accuracy
RF [26]	73.1	**LAB**	89.1
Linear SVM [26]	73.9	**FPFH**	74.5
NonLinear SVM [26]	74.8	**SIFT**	88.5
CKM Desc. [138]	90.8	**XYZ**	85.5
Kernel Desc. [146]	91.2	**LAB+FPFH**	93.6
HMP-RGBD [147]	92.8	**LAB+SIFT**	94.7
CNN [11]	94.1	**LAB+SIFT+FPFH**	95.7

Table 3.2: Instance recognition accuracy (%) on UW-RGBD. The algorithm names in bold indicate the variants of multi-domain pooling.

comparative algorithms and variants of our methods using different combinations of pooling domains (marked in bold type). We fix the filter scale as 0.03m to remove the effect of multi-scale filtering, in order to inspect how different pooling domains affect the classification performance. For color (LAB) and spatial (XYZ) domains, we only use single griding resolution of $5 \times 5 \times 5$. For gradient (SIFT) and 3D geometry (FPFH) domains, we apply 400 pooling regions. Similar to results shown in Table 3.1, we observe that the improvement of utilizing more diverse pooling domains. First, the LAB pooling domain individually outperforms SIFT and FPFH and is more efficient in computation. This decides that we apply LAB pooled features for extracting foreground during our two-stage semantic segmentation algorithm in Section 3.5. Second, SIFT and FPFH poolings are able to capture useful RGB-D patterns complimentary to the LAB-pooled feature while being less sensitive to illumination changes, which leads to the better performance of fine-grained multi-class classification compared with the single LAB pooling. The best result 95.7% is achieved by combined pooled features from three domains (i.e. LAB, FPFH and SIFT), which improves the current state-of-the-art [11] by 1.6%.

Algorithm	Acc.	Algorithm	Acc.
OUR-CVFH [136]	10.2	**XYZ-S-8**	31.2
ESF [135]	23.1	**LAB-S-8**	85.9
Kernel Descr. [146]	85.5	**ALL-S-8**	82.5
HMP-Depth [25]	35.1	**XYZ-M-8**	36.4
HMP-Color [25]	84.4	**LAB-M-8**	88.4
HMP-All [25]	80.8	**All-M-8**	84.6

Table 3.3: Testing accuracies (%) of different methods on BigBIRD. Variants of proposed method are marked in bold type.

3.3.3 BIGBIRD

We also tested our algorithm on the BigBIRD dataset [27]. This dataset contains 125 daily objects in which many object instances are very similar to each other. Each object has 600 Kinect-style RGB-D images covering five fixed viewing angles from 0 to 90 degrees [11]. As a result, the pose variation in BigBIRD is much larger than UW-RGBD in which object data is captured in three viewing angles of 30, 45 and 60 degrees. In turn, we adopt an architecture with a maximum of 8 stacked levels in both domains, in order to further analyze fine-grained pooling under a larger subset of 3D transformations. As far as we know, there is no evaluation metric for the object instance recognition on BigBIRD. Thus, we follow the similar experiment design in UW-RGBD to use sequences of the first, third and fifth viewing angles defined in BigBIRD for training and the remaining two for testing. We choose the state-of-the-art HMP[25], kernel descriptor [146] on UW-RGBD dataset and two rotationally invariant 3D descriptors OUR-CVFH [136] and ESF [135] for comparison.

[11] Though this dataset provides high-resolution color images and full 3D meshes, we only use the RGB-D images in this study.

Those methods are implemented with source codes provided by the authors[12] and the PCL library[13]. Parameters for all comparative methods are optimized by cross-validation on the first 30 objects. From Table 3.3, we observe our proposed architecture 'LAB-M-8' [14] achieves the highest recognition rate. Unlike the results in UW-RGBD, the combined domain is inferior to the color domain only. This is mainly because spatial pooling performs much worse than color pooling in both single and multiple filter scales.

For a more detailed analysis, we plot the recognition accuracies of each level (no stacking) in the color and spatial domains in Figure 3.5. Clearly, the testing accuracies achieved by the spatial domain drop dramatically in fine-grained levels while the color domain continuously boosts the accuracies. Also, the multi-scale filters still perform better than the single-scale ones, which coincides with the observation in UW-RGBD. Finally, we calculate the average probabilistic distances of Equation 3.5 between all pairs of object classes for pooled features using multi-scale filters. The solid (LAB-M-D) and dashed (XYZ-M-D) green lines show the average distances at each level in the LAB and XYZ domains, respectively. We can see that the distance metric derived in Equation 3.5 is able to describe the general trend of the recognition performance, which further validates the probabilistic framework in Section 3.2.3

[12] http://rgbd-dataset.cs.washington.edu/software.html

[13] http://pointclouds.org/

[14] Multiple scales of filters are specifically 0.02, 0.02, 0.02, 0.03, 0.03, 0.04, 0.04, 0.05, 0.05, 0.06 for levels from 8 to 1.

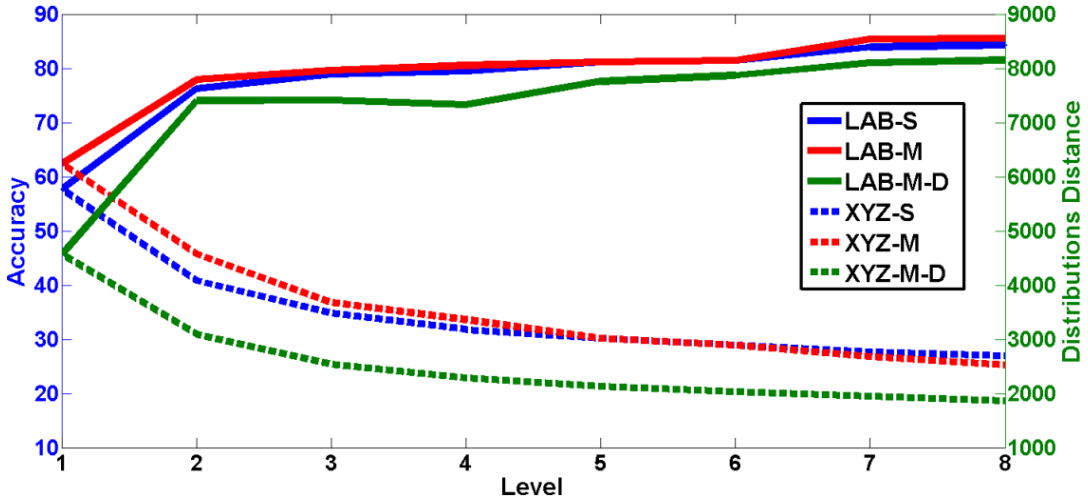


Figure 3.5: Classification accuracies at each level in pyramid and average distances (Equation 3.5) between all object classes in color and spatial pooling domains.

3.3.4 JHUIT-50

In the experiments on UW-RGBD and BigBIRD, testing data comes from sequences with fixed viewing angles. This constrained set of partial views may bias the evaluation of generalization performance towards a limited space in the entire viewing sphere, which is not desirable as a test for a realistic recognition scenario. In order to compensate for this drawback, we adopt two distinct collection procedures for training and testing data. On the training side, each object is placed on a turntable in increments of 7.2 degrees at three fixed camera viewing angles with 30, 45 and 60 degrees. This amounts to $\frac{360}{7.2} \times 3 = 150$ object views in total for training. For testing data, we manually move the camera around objects to sample another 150 random views of the object from the whole viewing sphere as the testing data. In this newly designed experiment setting, the testing data sampled from the full pose space contains larger pose variations than the previous two datasets. We

Algorithm	Acc.	Algorithm	Acc.
OUR-CVFH [136]	45,1	**XYZ-S-8**	75.5
ESF [135]	76.8	**LAB-S-8**	88.6
Kernel Descr. [146]	82.1	**ALL-S-8**	90.5
HMP-Depth [25]	41.1	**XYZ-M-8**	76.6
HMP-Color [25]	81.4	**LAB-M-8**	90.1
HMP-All [25]	74.6	**All-M-8**	91.2

Table 3.4: Testing accuracies (%) of different methods on IT.

deploy the same architecture with an 8 level pyramid used in BigBIRD on this dataset and the testing accuracies are reported in Table 3.4. We can clearly see that the experiment results on this dataset are similar to the previous two. First, color pooling and multi-scale filters are consistently superior to the spatial pooling and single-scale filters. Additionally, 'All-M-8' achieves the best result which significantly outperforms any others. Notice that spatial domain performs relatively better compared with the experiments on the BigBIRD dataset, though the pose variation is larger. This is mainly because the random testing views have overlaps with training views so that the spatial domain can positively contribute correct feature alignments for a subset of data.

3.3.5 Limitations

Although the proposed method achieves improvement over the current state-of-the-art on the aforementioned three datasets, two major limitations remain. First, fine-grained pooling in high levels (> 8) results in feature vectors with more than one million dimensions though it is sparse due to the soft-assignment encoder. This prevents more fine-grained implementations on

large-scale data. We could resolve this issue by using receptive field learning techniques [128, 129] to select a subset of pooling regions. Second, local features are not optimized for our new domain pooling. It is doable to back-propagate errors from these new pooling layers back to the convolutional layers within CNN and tune local filters to adapt to new pooling scheme. This has been empirically validated as one key reason for the success of CNN based on spatial pooling.

3.4 Sliding-Window Based Semantic Segmentation

In this Section, we review the first semantic segmentation algorithm based on multi-domain pooling reported in [37]. This is the preliminary version of the current semantic segmentation algorithm in Section 3.5.

3.4.1 Overview

Densely cluttered scenes are composed of multiple objects which are in close contact and heavily occlude each other. Existing semantic segmentation algorithms perform poorly on densely cluttered scenes mainly mainly due to the presence of objects with textureless surfaces, similar appearances and the difficulty of object instance segmentation. In the context of robotic manipulation, objects undergo frequent 3D transformation and appearances between different manufacturing objects and hand tools may be similar to each other. This motivates us to take advantage of multi-domain pooling to learn the object representation that is capable of discriminating between similar objects while preserving robustness to 3D transformation. We redesign the multi-domain

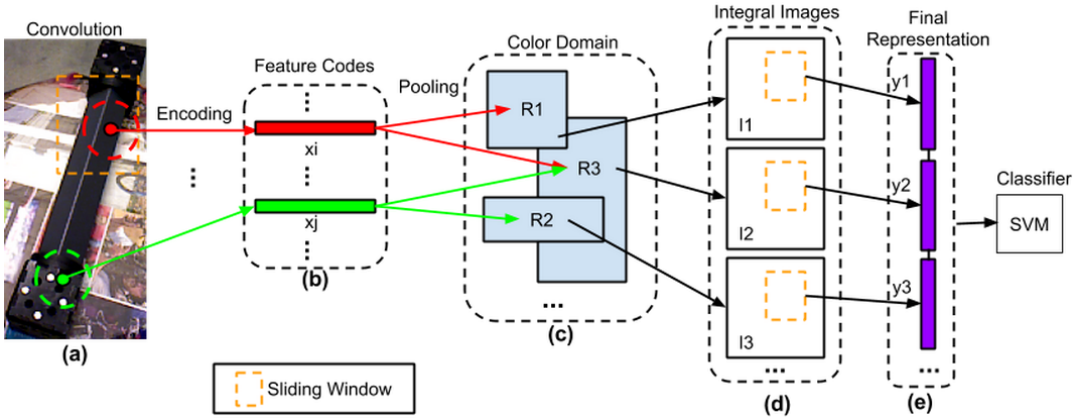


Figure 3.6: Illustration of the feature extraction for semantic segmentation, including from right to left: convolution of local feature codes (a→b), color pooling (b→c), integral image construction (c→d), and final feature concatenation (d→e).

pooling architecture above for the semantic segmentation.

There are three major challenges to achieve accurate and efficient semantic segmentation. First, robust and discriminative features need to be learned to distinguish different object classes, even for those with similar textureless appearances. Second, "mid-level" object models should be produced in order to handle clutter and occlusions caused by interactions among objects. Third, the state-of-the-art recognition techniques in large-scale setting typically do not operate at the time scales consistent with robot manipulation.

Here, we develop an algorithm for semantic segmentation based on the idea of color pooling in Section 3.2.4, but modified to make use of integral images to speed up the color pooling for sliding windows in the image domain. This enables the algorithm to perform efficient dense feature extraction in practice. We also detail how we exploit adaptive scales of sliding windows to achieve scale invariance for dense scene classification. The overview of the entire semantic segmentation pipeline is illustrated in Figure 3.6.

3.4.2 Efficient Computation via Integral Images

Integral images are often used for fast feature computation in real-time object detection such as [148]. We build the integral image structure for fast dense feature extraction. To do so, we first project each scene point cloud onto a 2D image using the camera's intrinsic parameters [15]. Suppose we obtain the local CSHOT feature vector x_i for each p_i. For each pooling region R_j, the corresponding integral image I_j is constructed as follows:

$$I_j(u,v) = \sum_i x_i \cdot \mathbf{1}(c_i \in R_j \wedge u_i \leq u \wedge v_i \leq v) \tag{3.13}$$

where (u,v) is the 2D coordinate of integral image and (u_i, v_i) is the projected 2D location of 3D point p_i in 3D point cloud.

The total complexity to construct all integral images is $O((K_d + K_c)WHm)$ where K_d and K_c are the number of codewords for color and depth components, respectively, and W and H are the width and height of integral images, respectively. Thus, with I_j, the pooled feature $y_j(B)$ for sliding window $B = \{u_l, v_l, u_r, v_r\}$ can be computed in $O(1)$:

$$y_j(B) = I_j(u_l, v_l) + I_j(u_r, v_r) - I_j(u_l, v_r) - I_j(u_r, v_l) \tag{3.14}$$

where (u_l, v_l) and (u_r, v_r) are 2D coordinates for top-left and bottom-right corners of window B on the projection of the 3D scene. Stages (c$\rightarrow$ d) and (d$\rightarrow$ e) in Fig. 3.6 show the process of integral image construction and pooled

[15] In our implementation, PrimeSense Carmine 1.08 depth sensor is used. We found no difference in performance between using default camera parameters and manual calibration.

feature extraction respectively.

3.4.3 Scale Invariant Modeling for Object Parts

Modeling object partial views from complete object segments does not account for missing object parts due to occlusion and outliers from background clutter. To overcome this, we train object models based on generic object parts randomly sampled from object segments at different viewpoints. In order to achieve the scale invariance for the learned models, all sampled parts are encompassed by a predefined fixed-size 3D bounding box $\mathcal{B}$. In turn, the sliding windows extracted for testing scene adopt scales which are consistent with $\mathcal{B}$. Specifically, the scale of the ith sliding window (w_i, h_i) with center (u_i, v_i) is equal to the scale of the projected bounding box $\mathcal{B}$ onto the same location:

$$(w_i, h_i) = \frac{\widetilde{f}}{z_i}(w_{\mathcal{B}}, h_{\mathcal{B}}) \tag{3.15}$$

where $(w_{\mathcal{B}}, h_{\mathcal{B}})$ is the predefined size in (x,y) coordinate of $\mathcal{B}$ in 3D and z_i is the depth corresponding to (u_i, v_i). $\widetilde{f}$ is the focal length of the camera. We note that object parts here do not necessarily have specific semantic correspondences. Next, we directly train a discriminative classification model using a linear SVM over object parts with semantic labels inherited from corresponding partial views.

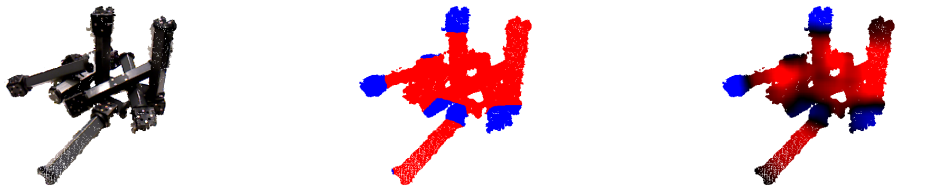
Given a new scene, we extract features with adaptive scales for all sliding windows on integral images. Each window is classified into one of the trained semantic classes and votes for all included 3D points. The final semantic label

of each 3D point is the one with the maximum votes.

3.4.4 Experiment on LN-66

We choose $0.03m$ as the radius for both normal estimation and CSHOT descriptor. The implementations of normal estimation and CSHOT come from PCL Library. Depth and color components in the raw CSHOT feature are decoupled into two feature vectors. Dictionaries with 200 codewords for each component are learned by hierarchical K-means. For the LAB pooling domain, we adopt a 4-level architecture where gridding over the entire domain at the k^{th} level is performed by equally dividing each channel of LAB into k bins. Therefore, in this 4-level architecture we have $100 = \sum_{k=1}^{4} k^3$ pooling regions. Pooled features in different levels and domains are concatenated as the final feature vector. Integral images are constructed with the size that is $\frac{1}{5}$ of the original RGB-D frame for efficiency. Sliding windows with step size of 1 pixel are extracted on integral images. Last, we capture object partial views under both fixed and random viewpoints as the training data for the SVM classifier in semantic segmentation. Specifically, three data sequences at fixed viewing angles of 30, 45 and 60 degrees as well as one random viewing sequence are captured. This follows the same procedure of data collection for JHUIT-50 dataset. In each partial view, we randomly sample 30 object patches encompassed by a predefined 3D bounding box with size $w_{\mathcal{B}} = h_{\mathcal{B}} = 0.03m$ (see details in Sec. 3.4.3). This size is also applied to compute the scale of sliding windows on integral images for testing examples.

We first evaluate our method on our new LN-66 dataset which contains 66



(a) Testing Scene. **(b)** Semantic Labels. **(c)** Confidence Map.

Figure 3.7: An example of semantic scene segmentation.

scenes with various complex configurations of the two "link" and "node" textureless objects shown in Figure 1.1b. We combine the training and testing sequences (corresponding to fixed and random viewpoints) of "link" and "node" objects in JHUIT-50 as the training data so that each object has 300 training samples. We note that our algorithm can easily be applied to scenes composed of more than 2 objects by simply adding more training classes in the semantic classification stage. The LN-66 dataset and the object training data are available at `http://cirl.lcsr.jhu.edu/jhu-visual-perception-datasets/`. An example testing scene is shown in Figure 3.7a. There are 6 to 10 example point clouds for each static scene from a fixed viewpoint, where each cloud is the average of ten raw RGB-D images. This gives a total of 614 testing examples across all scenes. In our dataset, the background has been removed from each example by RANSAC plane estimation and defining workspace limits in 3D space. Background subtraction can also be done with the semantic segmentation stage if object models are trained along with a background class. Therefore, the points in the remaining point cloud only belong to instances of the "link" or "node" objects. However, robust object detection and pose estimation are still challenging in such scenario due to similar appearances

between objects, clutter, occlusion and sensor noise. To quantitatively analyze our method, we manually label the groundtruth object poses for each scene and propagate them to all testing examples. Then the groundtruth poses are projected onto 2D to generate the groundtruth for the semantic segmentation at each frame.

The overall segmentation accuracy is measured as the average ratio of correctly labeled 3D points versus all in a testing scene point cloud. By running our semantic segmentation algorithm over all 614 testing frames, the average accuracy achieves as high as 91.2%. One example of semantic scene labeling is shown in Figure 3.7. The red and blue regions represent the "link" and "node" object classes, respectively. In Figure 3.7.c, we also show the confidence scores returned from the SVM classifier for each class. The brighter color in either red or blue indicates stronger confidence from the corresponding classifier. We could visually observe that the semantic segmentation obtains high classification accuracy.

3.5 Hierarchical Semantic Segmentation

The sliding-window based semantic segmentation introduced in the Section 3.4 performs poorly to capture the fine details of object boundary due to the nature of sliding window. In this section, we present a semantic segmentation algorithm based on supervoxels which is more robust to background clutter and more scalable than the approach in Section 3.4. Figure 3.8 shows an example of target scene for semantic segmentation algorithm in this Section. This type of scene is originated from the daily activity and contains the

Figure 3.8: An example of densely cluttered scene that is composed of objects with ambiguous appearances and in close contact.

background clutter that can not be easily segmented out.

3.5.1 Overview

The pipeline of our hierarchical semantic parsing algorithm is shown in Figure 3.9. We first present a hierarchy of region proposals which avoids relying on region merging heuristics used in most of the scene segmentation techniques [52, 46]. This region hierarchy explores a larger set of partial object regions that span from local to global patterns. The multi-domain pooled features (Section3.2) are efficiently propagated through the generic region hierarchy (Section 3.5.2) and the semantic labels of regions at all scales are combined for the robust semantic segmentation.

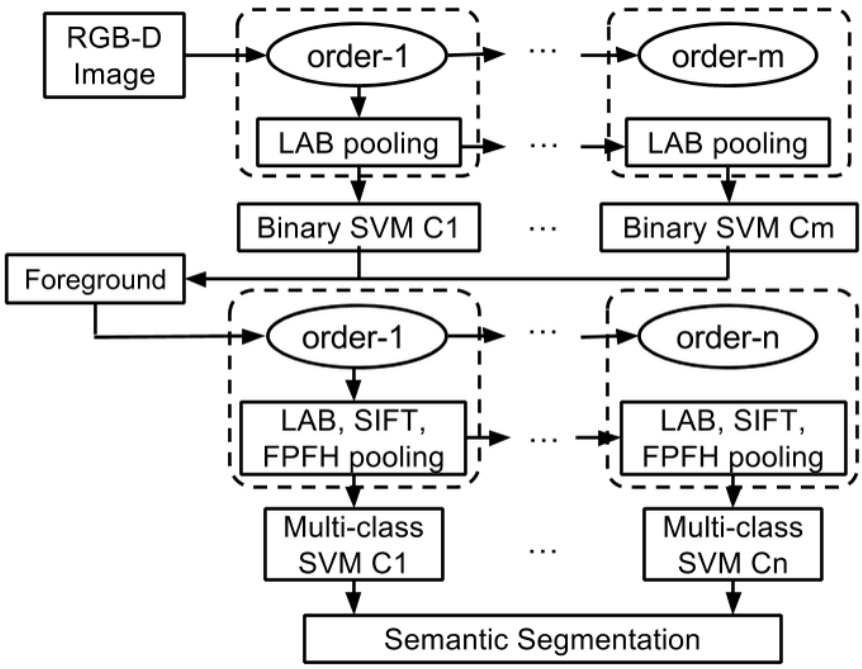


Figure 3.9: The flowchart of the hierarchical semantic parsing.

Our semantic segmentation algorithm is hierarchical along two dimensions. First, the semantic label for each supervoxel is evaluated and fused in various surrounding contexts based on the region hierarchy. Second, the semantic parsing is conducted in a hierarchical order, where the LAB-pooled feature is applied over the whole scene to extract foreground regions and the full multi-domain pooled feature from LAB, FPFH and SIFT is subsequently extracted on the predicted foreground for the multi-class object classification. This two-stage process is motivated by the fact that LAB pooling is more computationally efficient than SIFT and FPFH poolings since no additional local feature computation and nearest neighbor search among region seeds is conducted. We now describe each element of this algorithm in more detail.

3.5.2 Hierarchy of Multi-Scale Region Proposals

Consider an undirected graph $\mathcal{G} = \{V, E\}$ where vertexes and edges correspond to supervoxels and their adjacency connections returned from the 3D segmentation algorithm [149]. By accounting for color, depth and normal cues, the supervoxelization method [149] yields a sparsely connected graph which avoids the combinatorially growing number of regions at higher order sets. Subsequently, we define an order-k region set $\mathcal{O}^k = \{o_i^k\}$ in which each region o_i^k is a connected subgraph of $\mathcal{G}$ containing exactly k vertexes (i.e. k supervoxels that are path connected). In other words, o_i^k contains precisely k supervoxels (vertexes) where there exists at least one path between every pair of supervoxels. As a result, the $\mathcal{O}^1$ contains all raw individual supervoxels and $\mathcal{O}^2$ includes all connected pairs.

By induction, we can construct any higher order set $\mathcal{O}^k$ ($k > 1$) from $\mathcal{O}^{k-1}$ by growing each region $o_i^{k-1} \in O^{k-1}$ with all connected supervoxel neighbors one at a time and removing all region duplicates during this process. We summarize this generic region growing algorithm in Algorithm 1. Finally, the generic region hierarchy $\mathcal{H}$ is composed of all order sets $\mathcal{H} = \{\mathcal{O}^1, \cdots, \mathcal{O}^N\}$ with N orders. We note that region proposals in $\mathcal{H}$ are unique and may share common supervoxels, which is different from other methods such as [52].

3.5.3 Propagation of Multi-Domain Pooled Features

Consider two regions o_p and o_q from arbitrary order sets in the hierarchy $\mathcal{H}$. We compute the pooled feature $Y_{p+q}(r_{i,k}^t)$ in the pooling region $r_{i,k}^t \in R_k^t$ for

Algorithm 1 Generic Region Growing

Input $\mathcal{O}^{k-1}$ and graph $\mathcal{G} = \{V, E\}$
Initialize $\mathcal{O}^k = \emptyset$
for each region proposal $o_i^{k-1} \in \mathcal{O}^{k-1}$ **do**
  for each supervoxel $v_j \in o_i^{k-1}$ **do**
    for each neighbor v_k of v_j s.t. $< v_i, v_j > \in E$ **do**
      if $(v_k \notin o_i^{k-1}) \wedge (o_i^{k-1} \cup v_k \notin \mathcal{O}^k)$ **then**
        $\mathcal{O}^k = \mathcal{O}^k \cup (o_i^{k-1} \cup v_k)$
      end if
    end for
  end for
end for
Output $\mathcal{O}^k$

the combined region $o_p \cup o_q$ as follows:

$$Y_{p+q}(r_{i,k}^t) = \frac{\widehat{Y}_p(r_{i,k}^t) + \widehat{Y}_q(r_{i,k}^t)}{\|\widehat{Y}_p(r_{i,k}^t) + \widehat{Y}_q(r_{i,k}^t)\|_2} \tag{3.16}$$

where $\widehat{Y}_p(r_{i,k}^t)$ and $\widehat{Y}_q(r_{i,k}^t)$ are the unnormalized pooled features (defined in Equation 3.10) for o_p and o_q respectively. Finally, the concatenation of propagated features from all region seeds and domains forms the representation Y_{p+q} for $o_p \cup o_q$. Therefore, once we extract the pooled features for order-1 set (raw individual supervoxels), Equation 3.16 can be recursively applied to compute features for regions in higher order sets. Note that this feature propagation scheme does not work for spatially pooled features [9, 147, 11] because spatial pooling regions need to be reconstructed for each new merged region.

3.5.4 Two-Stage Semantic Segmentation

Suppose we have N object classes, the end goal of semantic segmentation is to classify each supervoxel into $N+1$ classes with one additional background class. We design a two-stage semantic parsing process that is shown in Figure 3.9. First, we build a shallow hierarchy $\mathcal{H}_f$ starting from order-1 set upon a given scene and only use LAB-pooled features to select supervoxels belonging to foreground class. Subsequently, a deeper hierarchy $\mathcal{H}_m$ is established upon foreground regions and more time-consuming SIFT and FPFH pooling are added along with the previously extracted LAB-pooled features to jointly classify different object classes. We deploy a deep hierarchy for $\mathcal{H}_m$ in order to mine discriminative features for large image regions for the fine-grained object classification.

Separate Support Vector Machines (SVM) are trained with respect to each order set $\mathcal{O}^k$ in both $\mathcal{H}_f$ and $\mathcal{H}_m$. We randomly sample region proposals from $\mathcal{O}^k$ in each object sample or background scene as the training data. All regions from object samples contribute as the positive data to train foreground/background SVM and no background region is involved for multi-class SVM training. At test time, each region proposal o_i^k will receive a vector of object class responses from the SVM for the kth order set. Finally, the class label of each supervoxel is inherited from the one with the maximum SVM score over all region proposals which contain it.

3.5.5 JHUScene-50

We evaluate the hierarchical semantic segmentation algorithm on JHUScene-50 [29] dataset. We assume that robots know the perception background in advance. Consequently, foreground/background SVM classifiers for all order sets in $\mathcal{H}_f$ are specifically trained for each environment from the associated background data. All multi-class SVMs for $\mathcal{H}_m$ are shared across the five indoor contexts.

Evaluation Metric. The performance of our semantic parsing algorithm is measured by the average precision and recall accuracies over all frames. Given a scene image, we denote S is the predicted segmentation map as the semantic partition over the image space. Similary, G is the groundtruth segmentation map. The precision and recall accuracies for each frame are calculated as the ratio of the overlap area between the semantic predictions by our algorithm and the groundtruth versus the overall labeling and groundtruth areas respectively:

$$precision = \frac{|S \cap G|}{|S|} \tag{3.17}$$

$$recall = \frac{|S \cap G|}{|G|} \tag{3.18}$$

We average precision and recall rates over all frames and all objects as the final accuracies.

Table 3.5 and 3.6 report the average precision and recall rates of the foreground and all ten objects in the five indoor environments. Three major observations follow. First, the recall rates of the foreground extraction are mostly

Scene	Precision / Recall		
	Foreground		
	$\mathcal{O}_1$	$\mathcal{O}_1$+$\mathcal{O}_2$	$\mathcal{O}_1$+$\mathcal{O}_2$+$\mathcal{O}_3$
Office	70.8 / 94.6	74.1 / 95.2	75.3 / 95.6
Labpod	64.9 / 90.0	70.6 / 90.4	71.8 / 91.7
Barrett	47.7 / 90.9	57.4 / 89.3	61.3 / 88.9
Box	63.8 / 89.5	71.3 / 90.5	71.8 / 92.2
Shelf	61.1 / 90.6	70.6 / 91.4	71.8 / 91.6
Overall	61.6 / 91.1	68.8 / 91.4	70.0 / 92.0

Table 3.5: Reported precision and recall of the foreground at different orders in $\mathcal{H}_f$ from 5 indoor environments.

above 90% across different environments, which means most of foreground regions are extracted for the subsequent multi-class classification. Though the precision of the foreground is relatively low compared with recall, it can be improved by applying object pose parsing module, such as the scene-model matching check via ObjRecRANSAC [4], to remove the false positives. Second, for both foreground and object, precision and recall rates are monotonically increasing in most of those environments[16] when higher order sets in $\mathcal{H}_f$ or $\mathcal{H}_m$ are deployed. This result indicates the effectiveness of our hierarchical semantic segmentation algorithm. Third, the object recall rates are mostly lower than the foreground recall by 20% $\sim$ 30%. This is mainly because the difficulty of correctly classifying ambiguous local and partial surfaces between objects with similar appearances. However, according to [4], ObjRecRANSAC [4] is able to estimate the correct pose from each partial view as long as more than 20% of the object surface is visible. As we show later, the 69.3% recall of object regions actually supports the pose estimation well on our dataset. Some qualitative results of the semantic segmentation are illustrated in Figure 3.10

[16]Only foreground recall slightly degrades in "barrett".

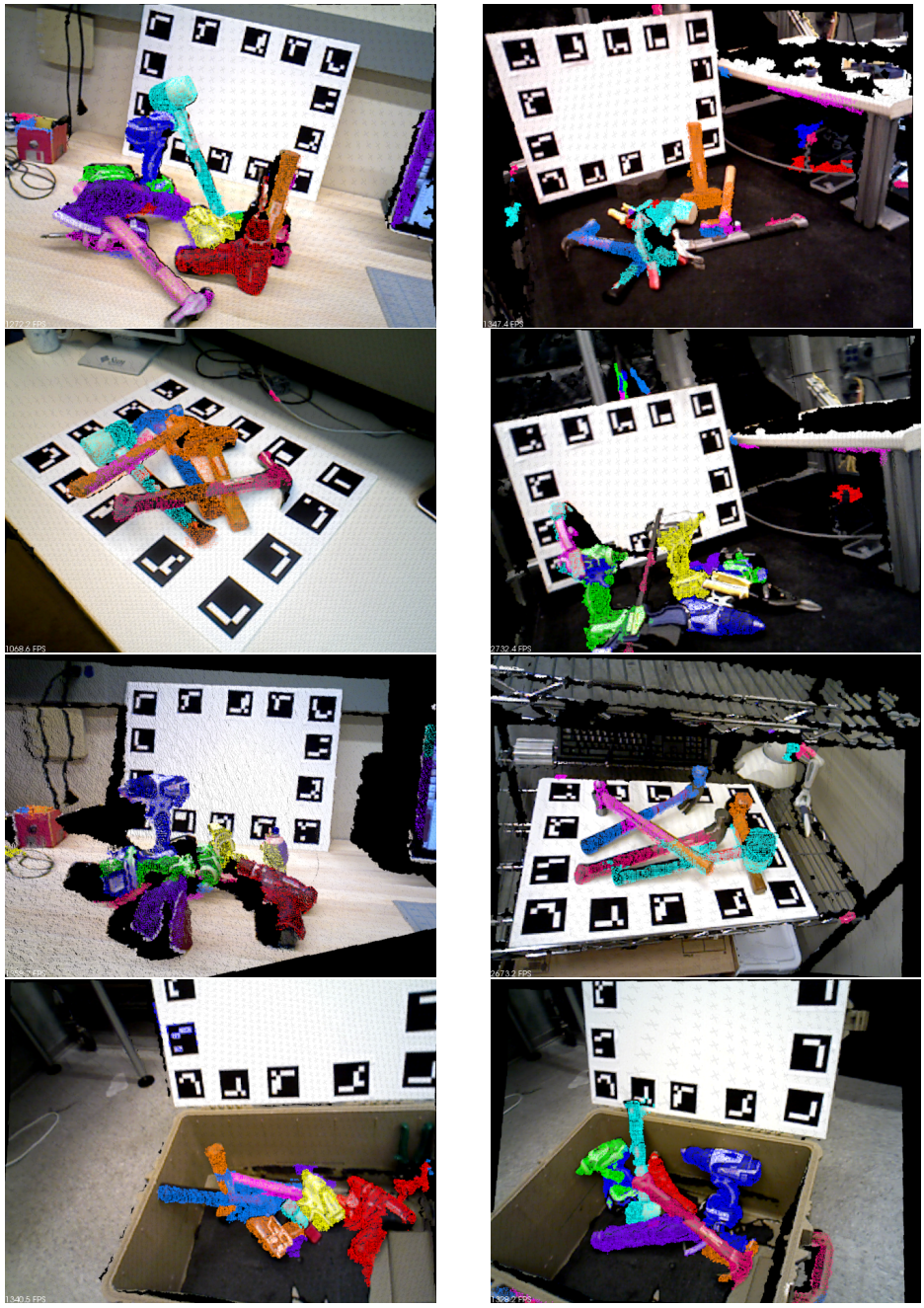


Figure 3.10: Visualization of the semantic segmentation result on JHUSecne-50. Each predicted semantic class is highlighted by a unique color.

Scene	Precision / Recall			
	All Objects			
	$\mathcal{O}_2$	$\mathcal{O}_2$+$\mathcal{O}_3$	$\mathcal{O}_2$+$\mathcal{O}_3$+$\mathcal{O}_4$	$\mathcal{O}_2$+$\mathcal{O}_3$+$\mathcal{O}_4$+$\mathcal{O}_5$
Office	70.3 / 69.3	72.3 / 73.2	74.2 / 72.1	74.5 / 73.6
Labpod	60.7 / 63.9	64.3 / 68.4	65.8 / 70.4	66.7 / 71.3
Barrett	55.5 / 53.6	57.9 / 57.3	58.7 / 58.8	59.7 / 59.0
Box	65.1 / 62.6	66.1 / 66.2	66.9 / 65.6	67.8 / 68.8
Shelf	72.4 / 68.7	74.7 / 72.3	75.5 / 73.5	75.9 / 74.1
Overall	64.5 / 63.6	66.8 / 67.5	67.3 / 68.8	67.6 / 69.3

Table 3.6: Reported precision and recall of all hand tool objects at different orders in $\mathcal{H}_m$ from 5 indoor environments.

3.6 Conclusion

In this chapter, we have presented a multi-domain pooling framework for learning object representation that is insensitive to common 3D transformations. The three main conclusions of this work are that: (1) a good fine-grained representation can be learned by fine-grained pooling within domains that are insensitive to object transformations; Using the natural capabilties of fine-grained pooling to construct more invariant learned representations, we are able to demonstrate superior object discrimination potential discrimination capability of a fine-grained pooling scheme could be fully exploited to build invariant fine-grained object representations within invariant domains; The combination of pooled features from LAB, SIFT and FPFH domains are robust to 3D transformations as well as illumination variations and achieve the state-of-the-art performance for the instance recognition; (2) filter responses over small-scale areas are preferred in fine-grained pooling while coarse-grained pooling being coupled with large-scale filters; (3) the spatial domain is much

less favorable than color domains towards learning representations that are invariant to 3D transformations, typically in the case of fine-grained pooling. We demonstrated that the proposed feature learning architecture significantly outperforms the current state-of-the-art on both public and self-collected datasets. While there is no question of developing discriminative 'deep' representations in a coarse-grained pooling setting, pooling simple local features within fine-grained levels of a domain that is insensitive to object transformations could significantly benefit object recognition as well.

In addition, we present a semantic segmentation algorithm to partition a scene into different object regions where object poses are estimated by a standard model registration method. The semantic segmentation is more accurate by fusing the predicted labels of region proposals at more diverse scales

We believe the theoretical pooling framework in this chapter can inspire a new design of feature learning architectures. For future work, not only can we explore new pooling domains with better invariance properties, but also new deep representations constructed beyond the spatial domain.

3.7 Appendix

In this appendix, we prove the following three theorems to support the derivations in Section 3.2.2 and 3.2.3.

Theorem 1

Theorem 1. *Given N random variables* $X_1, X_2, ..., X_N$, $E(\max_i X_i) \geq \max_i E(X_i)$ *and* $Var(\max_i X_i) \leq \sum_i Var(X_i)$.

Proof: The first conclusion $E(\max_i X_i) \geq \max_i E(X_i)$ directly follows from Jensen's inequality. Therefore, we focus on the proof for the second conclusion $\text{Var}(\max_i X_i) \leq \sum_i \text{Var}(X_i)$.

To begin, we show that given two independent random variables U,V that have the same distribution (i.e., $P(U) = P(V)$), $E(U - V)^2 = 2\text{Var}(U)$ holds with the fact that $E(X^2) = E(Y^2)$ and $E(XY) = E(X)E(Y) = [E(X)]^2$:

$$\begin{aligned} E(X - Y)^2 &= E(X^2 - 2XY + Y^2) \\ &= 2(E(X^2) - E(X)^2) \\ &= 2\text{Var}(X) \end{aligned} \tag{3.19}$$

Next, given N random variables $X_1, X_2, ..., X_N$ where each X_i has distribution $P(X_i)$, there always exists another N random variables $Y_1, Y_2, ..., Y_N$ subject to $P(Y_i) = P(X_i)$ and Y_i is independent from X_i (i.e., $P(X_i, Y_i) = P(X_i)P(Y_i)$). Suppose $\gamma \geq 0$, we denote an event A as $(\max_i X_i - \max_i Y_i)^2 > \gamma$ for random variables $\max_i X_i$ and $\max_i Y_i$. Note that $\max_i X_i$ is independent from $\max_i Y_i$. Additionally, we define another N events $B_1, B_2, ..., B_N$ where each B_i represents $(X_i - Y_i)^2 > \gamma$ and $1 \leq i \leq N$. As a result, when A occurs, at least $B_k \in \{B_1, ..., B_N\}$ is true where $k = \arg_i \max X_i$. Thus, the following

result can be deduced with the union bound:

$$\begin{aligned} P((\max_i X_i - \max_i Y_i)^2 > \gamma) = P(A) &\leq P(\cup_{i=1}^{N} B_i) \\ &\leq \sum_i P(B_i) \\ &\leq \sum_i P((X_i - Y_i)^2 > \gamma) \end{aligned} \tag{3.20}$$

Finally, based on Equation 3.19, the $\text{Var}(\max_i X_i) \leq \sum_i \text{Var}(X_i)$ is proved as follows:

$$\begin{aligned} \text{Var}(\max_i X_i) = \frac{1}{2} E(\max_i X_i - \max_i Y_i)^2 &= \frac{1}{2} \int P((\max_i X_i - \max_i Y_i)^2 > \gamma) d\gamma \\ &\leq \frac{1}{2} \sum_{i=1}^{N} \int P((X_i - Y_i)^2 > \gamma) d\gamma \\ &\leq \sum_{i=1}^{N} \frac{1}{2} E((X_i - Y_i)^2) \\ &\leq \sum_{i=1}^{N} \text{Var}(X_i) \end{aligned} \tag{3.21}$$

Therefore, we have $\text{Var}(\max_i X_i) \leq \sum_{i=1}^{N} \text{Var}(X_i)$. Note that this theorem allows $X_1, X_2, ..., X_N$ to be dependent from each other.

Theorem 2

Theorem 2. *Given N random variables $X_1, X_2, ..., X_N$, $Var(\frac{1}{N} \sum_i X_i) \leq \sum_i Var(X_i)$.*

Proof: Given N random variables $X_1, X_2, ..., X_N$ where each X_i has distribution $P(X_i)$, there always exists another N random variables $Y_1, Y_2, ..., Y_N$ subject to $P(Y_i) = P(X_i)$ and Y_i is independent from X_i (i.e., $P(X_i, Y_i) = P(X_i)P(Y_i)$). Suppose $\gamma \geq 0$, we denote an event A as $(\frac{1}{N}\sum_i X_i - \frac{1}{N}\sum_i Y_i)^2 > \gamma$ for random variables $\frac{1}{N}\sum_i X_i$ and $\frac{1}{N}\sum_i Y_i$. Furthermore, we define another N events $B_1, B_2, ..., B_N$ where each B_i represents $(X_i - Y_i)^2 > \gamma$ and $1 \leq i \leq N$.

Next, we prove by contradiction that if A is true, at least one B_i is true. Specifically, we assume that when A is true, all B_i are false (i.e., $|X_i - Y_i| \leq \sqrt{\gamma}$ for $1 \leq i \leq N$). Then, with the triangle inequality, we get the following result:

$$\left|\frac{1}{N}\sum_i X_i - \frac{1}{N}\sum_i Y_i\right| \leq \frac{1}{N}\sum_{i=1}^{N}|X_i - Y_i| \leq \sqrt{\gamma} \tag{3.22}$$

As a consequence, if all B_i are false, $(\frac{1}{N}\sum_i X_i - \frac{1}{N}\sum_i Y_i)^2 \leq \gamma$ follows, which contradicts that A is true. Therefore, this shows that at least one B_i needs to be true if A occurs. With the union bound, we can get:

$$\begin{aligned} P((\frac{1}{N}\sum_i X_i - \frac{1}{N}\sum_i Y_i)^2 > \gamma) = P(A) &\leq P(\cup_{i=1}^{N} B_i) \\ &\leq \sum_i P(B_i) \\ &\leq \sum_i P((X_i - Y_i)^2 > \gamma) \end{aligned} \tag{3.23}$$

Finally, analogous to Equation 3.21, we can get $\text{Var}(\frac{1}{N}\sum_i X_i) \leq \sum_i \text{Var}(X_i)$ by using Equation 3.23. Note that this theorem allows $X_1, X_2, ..., X_N$ to be dependent.

Theorem 3

The following theorem is used to explain: $\text{Var}(x_{jk}^p) \propto \text{Var}(d_k|s_j, o_p)$ and $\text{Var}(x_{jk}^p) \propto \text{Var}(s_j|d_k, o_p)$.

Theorem 3. *Given two independent random variables X and Y, $Var(XY)$ are positively proportional to $Var(X)$ and $Var(Y)$. That is, $Var(XY) \propto Var(X)$ and $Var(XY) \propto Var(Y)$ if $P(X,Y) = P(X)P(Y)$.*

Proof: We know that, for any two independent variables X and Y, $E(XY) = E(X)E(Y)$ and $E(X^2Y^2) = E(X^2)E(Y^2)$. Therefore, $\text{Var}(XY) \propto \text{Var}(X)$ and $\text{Var}(XY) \propto \text{Var}(Y)$ follow from:

$$\begin{aligned}
\text{Var}(XY) &= E[(XY)^2] - [E(XY)]^2 \\
&= E(X)^2E(Y)^2 - [E(X)]^2[E(Y)]^2 \\
&= [E(X)]^2\text{Var}(Y) + [E(Y)]^2\text{Var}(X) + \text{Var}(X)\text{Var}(Y)
\end{aligned} \tag{3.24}$$

Next, we use the theorem above to substantiate the variance statement associated with Equation 3.6 and 3.7 in the Section 3.2.3. We already define $x_{jk}^p = (s_j, d_k)|o_p$ as the activation strength of d_k at s_j given object o_p. Therefore, x_{jk}^p can be decomposed into the following two forms: $x_{jk}^p = u_{jk}^p v_j^p$ and $x_{jk}^p = w_{jk}^p q_k^p$, where $u_{jk}^p = (d_k|s_j, o_p)$, $v_j^p = (s_j|o_p)$ and $w_{jk}^p = (s_j|d_k, o_p)$, $q_k^p = (d_k|o_p)$. More specifically, u_{jk}^p is the activation score of d_k at s_j and v_j^p is a 0-1 variable indicating whether object o_p occupies s_j. Also, w_{jk}^p is a 0-1 variable indicating whether response value d_k at s_j from the k^{th} filter (we abuse the notation d_k to indicate the continuous response value of the kth filter) and $q_k^p = d_k|o_p$

represents the response value d_k of the kth filter given o_p. It is clear that u_{jk}^p is independent from v_j^p and w_{jk}^p is independent from q_k^p. With Theorem 3, we can derive $\text{Var}(x_{jk}^p) \propto \text{Var}(d_k|s_j, o_p)$ and $\text{Var}(x_{jk}^p) \propto \text{Var}(s_j|d_k, o_p)$.

Chapter 4

Deep Supervision With Intermediate Concepts

In this chapter, we present a generalized deep supervision framework for improving generalization capability over standard single-task or multi-task supervision schemes. We first motivate our approach and outline the critical innovations in Section 4.1. Subsequently, we present the core methodology in Section 4.3, which reveals the theoretical insights and defines the generalized deep supervision algorithm. In Section 4.4.1, we show how to exploit off-the-shelf synthetic simulation pipeline to provide training data. Finally, we show two applications of 2D/3D keypoint localization and image classification in the 4.4 and 4.5.

4.1 Motivation and Overview

Our visual world is rich in structural regularity. All of humanity together has never seen most realizations of even a 10x10 image, yet we are able to

comprehend our surroundings from our visual inputs remarkably well. Studies in perception show that the human visual system imposes structure to reason about stimuli[150]. Consequently, early work in computer vision studied perceptual organization as a fundamental precept for recognition and reconstruction [151, 152]. However, algorithms designed on these principles relied on hand-crafted features (e.g. corners or edges) and hard-coded rules (e.g. junctions or parallelism) to hierarchically reason about abstract concepts such as shape [153, 154]. Such approaches suffered from limitations in the face of real-world complexities. In contrast, convolutional neural networks (CNNs), as end-to-end learning machines, ignore inherent perceptual structures encoded by task-related intermediate concepts and attempt to directly map from input to the label space.

Abu-Mostafa [155] proposed the use of "hints" as a middle ground, where a task-related hint derived from prior domain knowledge regularizes the training of deep neural networks by either constraining the parameter space or generating more training data. However, it is still vague how to apply this idea to assist generic vision task like image classification or 3D structure understanding. In this work, we revisit and extend this idea by exploring a specific type of hint which we refer to as an "intermediate concept" which encodes a sub-goal to achieve the main task of interest. Such intermediate concepts commonly emerge when we concretely define a generic vision task. For instance, knowing object orientation is a prerequisite to correctly infer object part visibility which in turn constrains the 3D locations of semantic object parts. We present a generic learning architecture where intermediate

concepts sequentially supervise hidden layers of a deep neural network to learn a specific inference sequence for predicting a final task.

4.1.1 Motivating Example

To motivate the idea of supervising intermediate concepts, let us look at a toy example as follows. Consider a network with 2 layers: $y = \sigma(w_2\sigma(w_1x + b_1) + b_2)$ where σ is ReLU activation $\sigma(x) = \max(x, 0)$. Provided that the true model for a phenomenon is $(w_1, w_2, b_1, b_2) = (3, 1, -2, -7)$ and the training data $\{(x, y)\}$ is $\{(1, 0), (2, 0), (3, 0)\}$. A learning algorithm may obtain a different model $(w_1, w_2, b_1, b_2) = (1, 3, -1, -10)$ which still achieves zero loss over training data but fails to generalize to the case when $x = 4$ or 5. However, if we have additional cues that tell us the value of intermediate layer activations, $\sigma(w_1x + b_1)$ for each (x, y), we can achieve better generalization. For example, suppose we have training examples with an additional intermediate cue $\{(x, y', y)\} = \{(1, 0, 0), (2, 0, 0), (3, 1, 0)\}$ where $y' = \sigma(w_1x + b_1)$. We find that the incorrect solution above that works for $\{x, y\}$ is removed because it does not agree with $\{x, y', y\}$. While simple, this example illustrates that deep supervision with intermediate concepts can regularize network training and reduce overfitting.

4.2 Related Work

We present a deep supervision scheme with intermediate concepts for deep neural networks. Our method is similar in spirit to multi-task learning architecture where multiple task-related concepts jointly supervise a deep neural

network along with the main task at the last layer. We review related work on deep learning architectures in the following:

Multi-task Learning. In neural networks, multi-task learning architectures exploit multiple task-related concepts to jointly supervise a network at the last layer. Caruana [156] empirically demonstrates its advantage over a single-task neural architecture on various learning problems. Recently, multi-task learning has been applied to a number of vision tasks including face landmark detection [157] and viewpoint estimation [65]. Hierarchy and Exclusion (HEX) graph [158] is proposed to capture hierarchical relationships among object attributes for improved image classification. In addition, some theories [159, 160] attempt to investigate how shared hidden layers reduce required training data by jointly learning multiple tasks. However, to our knowledge, no study has been conducted on quantifying the performance boost to a main task. It is also unclear whether a design choice meets the assumption of conducive task relationships used in these theories. This may explain that some task combinations for multi-task networks yield worse performance compared with single-task networks [156].

Deep Supervision. Deeply Supervised Nets (DSN) [35] uses a single task label to supervise the hidden layers of a CNN, speeding up convergence and addressing the vanishing gradient problem. However, DSN assumes that optimal local filters at shallow layers are building blocks for optimal global filters at deep layers, which is probably not true for a complex task. Recently a two-level supervision is proposed [161] for counting objects in binary images. One hidden layer is hard-coded to output object detection responses at fixed

image locations. This work can be seen as a preliminary study to leverage task-related cues that assist the final task by deep supervision. We advance this idea further to a more general setting for deep learning without hard-coded internal representations.

Hierarchical Supervision. Hierarchy and Exclusion (HEX) graphs [158] are proposed as a formalism to capture hierarchical relationships among classification labels as a Conditional Random Field model; evaluated on ImageNet classification. The work demonstrates significantly improved object classification by hand-modeling various label relationships. Our application focus here is on 3D object parsing, yet we expect that our deep supervision approach maybe a step towards learning such dense label hierarchies implicitly inside deep neural networks.

Domain Adaptation. A number of recent works [89, 90, 91] propose to transfer knowledge learned on a labeled "source" dataset, by matching activation statistics of intermediate CNN layers against an unlabeled "target" dataset and employing Maximum Mean Discrepancy (MMD) losses. These methods have so far been applied to relatively simple datasets, such as MNIST digits and Caltech 101. Our approach explores an orthogonal route to enforce knowledge transfer by employing losses of intermediate tasks. MMD losses can be also leveraged for our method but this exploration lies outside the scope of the present paper.

Generalization of CNNs from synthetic data to real images has been considered in recent works for multi-object tracking [162], optical flow estimation [163], semantic segmentation [164] and viewpoint estimation [65]. The

strategy of these methods is to generate a large amount of synthetic data to compensate for the domain gap. Our approach goes further by providing explicit intermediate supervision signals, which regularize the network to focus on task-relevant regularities as opposed to irrelevant coincidental regularities in synthetic training data.

Graphics Modeling. Recently, CNN has been adapted to simulate the graphics rendering by [75] where a compact set of object properties can be mapped to a rendered image describing that object. Kulkarni and et al.[94] take one step further by adding an encoder module to simulate the inverse graphics process in order to connect the real image and display more realistic view. Multi-view CNN [76] shares the same spirit to produce the RGB and depth images given a single image and a pose parameter. However, their objective is to generate high quality synthetic images, which is different from ours. The more recent work [77] applies the CNN to output graphics parameters from the last layer in order for 2D/3D object completion. But it is unknown how it performs on real data and does not conduct the deep supervision as we do.

4.3 Methodology

In this section, we introduce a novel CNN architecture with deep supervision. Our approach draws inspiration from Deeply Supervised Nets (DSN) [35]. DSN supervises each layer by the main task label to accelerate training convergence. Our method differs from DSN in that we sequentially apply deep supervision on intermediate concepts intrinsic to the ultimate task, in order to regularize the network for better generalization. We employ this enhanced

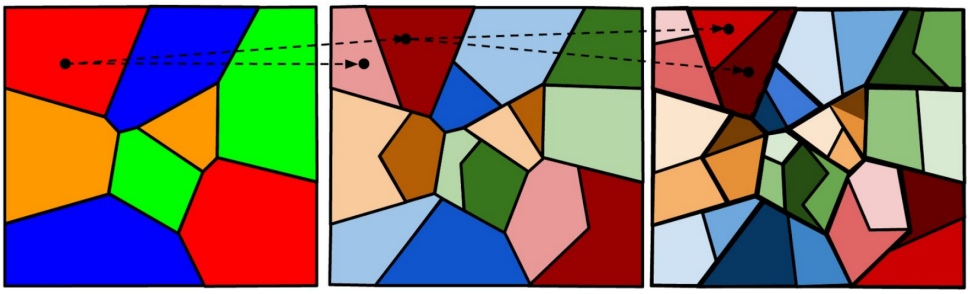


Figure 4.1: Illustration of a concept hierarchy with three concepts $\mathcal{Y} = \{y_1, y_2, y_3\}$ on 2D input space. Black arrows indicate the finer decomposition within the previous concept in the hierarchy. Each color represents one individual class defined by the concept.

generalization ability to transfer knowledge from richly annotated synthetic data to the domain of real images.

In the following, we formalize the notion of intermediate concept in Section 4.3.1, introduce our supervision approach which exploits intermediate concepts in Section 4.3.2, and discuss the improved generalization of deep supervision in Section 4.3.3.

4.3.1 Intermediate Concepts

In many computer vision problems, we posit that the main prediction task depends on some task-related concepts which contribute to the ultimate task. For example, the knowledge of object pose imposes a certain distribution on 3D locations of object semantic parts, which constrains the prediction space for 3D parts. In the following, we formally define such task-related intermediate concepts.

Let us consider a supervised learning task to predict y_m from x. We have a

training set $S = \{(x, (y_1, \cdots, y_m))\}$ sampled from an unknown distribution $\mathcal{D}$. Set S contains training samples with $m-1$ intermediate concepts $y_1, \cdots, y_{m-1}$ to the ultimate task y_m. Each training tuple consists of multiple task labels: $(y_1, \cdots, y_m)$. Without the loss of generality, we focus on analyzing the i-th concept y_i in the following, where $1 < i \leq m$. y_{i-k} is regarded as an intermediate concept to estimate y_i where $k > 0$ and $i - k > 0$. Intuitively, knowledge of y_{i-k} constrains the solution space of y_i, as in our simple example shown above.

In this work, we seek to exploit a type of concept relationship termed as "necessary condition". We define y_{i-k} as an ϵ-error necessary condition of y_i ($0 \leq \epsilon \leq 1$) if y_{i-k} and y_i for any sample $(x, (y_{i-k}, y_i)) \sim \mathcal{D}$ satisfy:

$$\forall c, \ \max \mathrm{P}(y_{i-k} \mid y_i = c) \geq 1 - \epsilon \tag{4.1}$$

where $0 \leq \epsilon \leq 1$. If ϵ is small, we have a high probability to determine the value of y_{i-k} given any possible value of y_i. For convenience, we abuse the notation such that y_i indicates not only the random variable but also its realization. Note that the value of y_i (or y_{i-k}) can be either discrete or continuous.

When $\epsilon = 0$, we call y_{i-k} a strict necessary condition of y_i. In other words, we can obtain a deterministic function T which maps y_i to y_{i-k}: $y_{i-k} = T(y_i)$. In general, we are unable to find an inverse function T' that maps y_{i-k} to y_i because multiple y_i may map to the same y_{i-k}. In the context of multi-class classification where task y_i and y_{i-k} both contain discrete class labels, task y_i induces a finer partition over the input space $\mathcal{X} = \{x\}$ than task y_{i-k}

by further partitioning each class in y_{i-k}. Figure 4.1 illustrates a fictitious example of hierarchical partitioning over 2D input space created by three intermediate concepts $\{y_1, y_2, y_3\}$ with $\epsilon = 0$. As we can see in Figure 4.1, a sequence of intermediate concepts hierarchically decompose the input space from coarse to fine granularity. When $\epsilon > 0$, error ϵ captures the uncertainty about the strict necessary condition between two concepts. The corresponding geometric interpretation becomes that partitions induced by y_j are mixed by data from other classes. The more such mixture, the larger ϵ. Concretely, we denote ϵ-error concept hierarchy as $\mathcal{Y}_\epsilon = (y_1, \cdots, y_m)$ where y_{i-k} is an ϵ-error necessary condition of y_i for all $i > 1$.

In many vision problems, we should be intuitively able to find concept hierarchies with small ϵ. As mentioned above, non-overlapping coarse-grained class labels constitute natural strict necessary conditions for a fine-grained classification task. In addition, 6-DoF object pose and keypoint visibility are both strict necessary conditions for 3D object keypoint location because the former can be unambiguously determined by the later.

4.3.2 Algorithm

Given a concept hierarchy $\mathcal{Y}_\epsilon$ and the corresponding training set S, we formulate a new deeply supervised architecture to jointly learn the main task along with its intermediate concepts. Consider a multi-layer convolutional neural network with N hidden layers that receives input x and outputs m predictions for $y_1, \cdots, y_m$. The i-th concept y_i is applied to supervise the intermediate hidden layer at depth d_i by adding a side output branch at d_i-th hidden layer. We

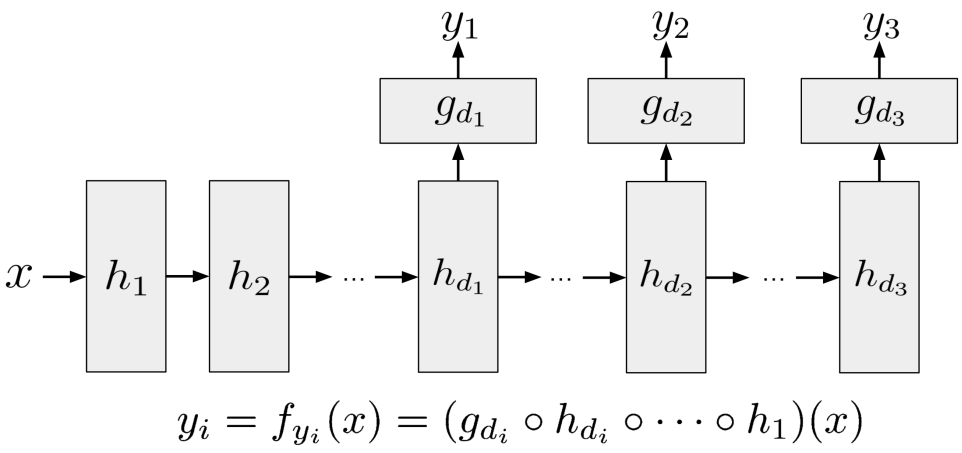


Figure 4.2: Schematic diagram of deep supervision framework.

denote the function of k-th hidden layer as $h_k(x, W_k)$ with the parameter W_k. The output branch at depth d_i constructs a function $g_{d_i}(\cdot, V_{d_i})$ with parameter V_{d_i}. Further, we denote f_{y_i} as the function for predicting concept y_i such that $f_{y_i} = g_{d_i} \circ h_{d_i} \circ \cdots \circ h_1$. Note that g_{d_i} outputs a prediction for y_i by applying some decision rule over the output of last layer. Figure 4.2 shows a schematic diagram of the deep supervision framework. In Section 4.4, we concretely instantiate each h_k as a convolutional layer followed by batch normalization and ReLU layers, and each g_k as global average pooling followed by fully connected layers. However, we emphasize that our algorithm is not limited to this particular layer configuration.

Thus, we formulate the following objective function to encapsulate these ideas:

$$W^*, V^* = \underset{W,V}{\operatorname{argmin}} \sum_{(x,\{y_i\})\in S} \sum_{i=1}^{m} \lambda_i l_i(y_i, f_{y_i}(x\,; W_{1:d_i}, V_{d_i})) \tag{4.2}$$

where $W_{1:d_i} = \{W_1, \cdots, W_{d_i}\}$, $W = \cup_i W_{1:d_i}$ and $V = \{V_{d_1}, \cdots, V_{d_m}\}$. In addition, l_i is the loss for task y_i scaled by the loss weight λ_i. We optimize

Notation	Meaning
y_i	The i-th concept
y_{i-k}	The intermediate concept of y_i
d_i	The supervision depth of y_i
f_{y_i}	A function predicts y_i given input x
$R(f_{y_i})$	True risk of f_{y_i}
$R_S(f_{y_i})$	Empirical risk of f_{y_i} given a training set S
$\mathcal{H}_{y_i}$	A set of f_{y_i} with low empirical risk
$\mathcal{F}_{y_i}$	A set of f_{y_i} with low empirical and true risk
P_{y_i}	Generalization probability of y_i
$\mathcal{H}_{y_i\|y_{i-k}}$	A subset of $\mathcal{H}_{y_i}$ achieves low empirical risk on y_{i-k}
$\mathcal{F}_{y_i\|y_{i-k}}$	A subset of $\mathcal{F}_{y_i}$ achieves low empirical risk on y_{i-k}
$\mathrm{P}_{y_i\|y_{i-k}}$	Generalization probability y_i constrained by y_{i-k}

Table 4.1: Notation table.

Equation 4.2 over S by simultaneously backpropagating the loss of each supervisory signal all the way back to the first layer.

We note that Equation 4.2 is a generic supervision framework which represents many existing supervision schemes. For example, the standard CNN with a single task supervision is a special case when $m = 1$. Additionally, the multi-task learning [156] places all supervision on the last hidden layer: $d_i = N$ for all i. DSN[35] framework is obtained when $m = N$ and $y_i = y_m$ for all i. In this work, we propose to apply m different concepts $\{y_i\}$ in a concept hierarchy $\mathcal{Y}_\epsilon$ at locations with growing depths: $d_{i-k} < d_i$ where $k > 0$ and $i - k > 0$. In the Section 4.3.3, we will formulate a probabilistic framework which demonstrate that this proposed supervision scheme obtains better generalization than other standard supervision methods.

4.3.3 Generalization Analysis

In this section, we present a generalization metric and subsequently show that deep supervision with intermediate concepts could improve the generalization of a deep neural network with respect to this metric, compared to other standard supervision methods. We also discuss the limitations of this analysis. For clarity, we summarize our notation in Table 4.1.

4.3.3.1 Generalization Metric

Deep neural networks are function approximators that learn mappings from an input space x to an output space y. For a network with a fixed structure, there usually exists a set of functions $\mathcal{H}$ (equivalently a set of parameters) where each element $f \in \mathcal{H}$ achieves a low empirical loss on a training set S. In the following, we define a generalization metric to measure the probability that a function $f \in \mathcal{H}$ is a "true" solution for a supervised learning task. In the case of multiple concepts $\{y_1, \cdots, y_m\}$, we assume that a "true" solution for y_m also exactly satisfies all its intermediate tasks from y_1 to y_{m-1}, as suggested by the toy example earlier.

Recall that f_{y_i} represents the function composed by the first d_i hidden layers and an output branch for predicting concept y_i. The true risk $R(f_{y_i})$ is defined based on random variables x and y_i where $(x, y_i) \sim D$:

$$R(f_{y_i}) = \mathbb{E}\left[l_i(f_{y_i}(x), y_i) \right] \tag{4.3}$$

Given a training set S, the empirical risk $R_S(f_{y_i})$ of f_{y_i} is:

$$R_S(f_{y_i}) = \frac{1}{|S|} \sum_{(x,y_i)\in S} l_i(f_{y_i}(x), y_i) \tag{4.4}$$

Given limited training data S, a deep neural network is optimized to find a solution f_{y_i} with low empirical loss. We consider empirical loss to be "low" when $R_S(f_{y_i}) < \delta$. δ is the risk threshold which indicates "good" performance for a task. Next, we define the function set $\mathcal{H}_{y_i}$ in which each function achieves low empirical risk:

$$\mathcal{H}_{y_i} = \{f_{y_i} \mid R_S(f_{y_i}) < \delta\} \tag{4.5}$$

Similarly, we also define the function set $\mathcal{F}_{y_i}$ where each function achieves risks less than δ for both $R(f_{y_i})$ and $R_S(f_{y_i})$:

$$\mathcal{F}_{y_i} = \{f_{y_i} \mid R_S(f_{y_i}) < \delta \ \wedge \ R(f_{y_i}) < \delta\} \tag{4.6}$$

By definition, we know $\mathcal{F}_{y_i} \subseteq \mathcal{H}_{y_i}$. Given a training set and network structure, the generalization capability of the outcome of network training depends upon the likelihood that $f_{y_i} \in \mathcal{H}_{y_i}$ is also a member of $\mathcal{F}_{y_i}$. Such an f_{y_i} achieves the true risk at least as good as the empirical risk.

We consider f_{y_i} to be a random variable as it is the outcome of a stochastic optimization process such as stochastic gradient descent. We assume that the optimization algorithm is unbiased within $\mathcal{H}_{y_i}$, such that apriori probability of converging to any $f_{y_i} \in \mathcal{H}_{y_i}$ is uniformly distributed. We formalize a generalization metric for a CNN for predicting y_i by defining a probability

measure P_{y_i} based on the function sets $\mathcal{F}_{y_i}$ and $\mathcal{H}_{y_i}$:

$$\begin{aligned}\mathrm{P}_{y_i} &= \mathrm{P}(R(f_{y_i}) < \delta \mid R_S(f_{y_i}) < \delta) \\ &= \begin{cases} \frac{\mu(\mathcal{F}_{y_i})}{\mu(\mathcal{H}_{y_i})} & : \mathcal{H}_{y_i} \neq \emptyset \\ 0 & : \mathcal{H}_{y_i} = \emptyset \end{cases}\end{aligned} \tag{4.7}$$

where $\mu(A)$ is the Lebesgue measure [165] of set A indicating the "volume" or "size" of set A [1]. For any f_{y_i} with risk (or empirical risk) lower than δ, there exists an epsilon ball surrounding its corresponding parameter W which also achieves the risk less than δ. This indicates that each function set $\mathcal{F}_{y_i}$ or $\mathcal{H}_{y_i}$ contains a union of subsets of $\mathbb{R}^n$ which correspond to the neighborhood of some local minimum. Thus, $\mathcal{H}_{y_i}$ and $\mathcal{F}_{y_i}$ are Lebesgue measurable and $\mu(\mathcal{F}_{y_i}) \leq \mu(\mathcal{H}_{y_i})$ due to $\mathcal{F}_{y_i} \subseteq \mathcal{H}_{y_i}$. Moreover, $\mu(\mathcal{F}_{y_i}) \leq \mu(\mathcal{H}_{y_i})$ due to $\mathcal{F}_{y_i} \subseteq \mathcal{H}_{y_i}$. The equality $\mu(\mathcal{F}_{y_i}) = \mu(\mathcal{H}_{y_i})$ is achieved when $\mathcal{F}_{y_i} = \mathcal{H}_{y_i}$. It follows that the higher the P_{y_i}, the better the generalization.

When an intermediate concept y_{i-k} of y_i is available, we insert one output branch $g_{d_{i-k}}$ at depth d_{i-k} of CNN to predict y_{i-k}. Then, our deep supervision algorithm in Section 4.3.2 aims to minimize empirical risk on both y_{i-k} and y_i. Recall that $f_{y_i} = g_{d_i} \circ f_{d_i} \circ \cdots \circ f_1$. As a consequence, f_{y_i} does not contain any output branch $g_{d_{i-k}}$ for the intermediate concept y_{i-k}. However, we note that f_{y_i} shares some hidden layers with $f_{y_{i-k}}$. Similar to P_{y_i}, we can define the generalization probability $\mathrm{P}_{y_i|y_{i-k}}$ of f_{y_i} given the supervision of its

[1] Each function f_{y_i} has a one-to-one mapping to a parameter W in $\mathbb{R}^n$ where n is the dimension of the parameter. We know that any subset of $\mathbb{R}^n$ is Lebesgue measurable.

intermediate concept y_{i-k}:

$$\begin{aligned} \mathrm{P}_{y_i|y_{i-k}} &= \mathrm{P}(R(f_{y_i}) < \delta \mid R_S(f_{y_i}) < \delta, R_S(f_{y_{i-k}}) < \delta') \\ &= \begin{cases} \frac{\mu(\mathcal{F}_{y_i|y_{i-k}})}{\mu(\mathcal{H}_{y_i|y_{i-k}})} & : \mathcal{H}_{y_i|y_{i-k}} \neq \emptyset \\ 0 & : \mathcal{H}_{y_i|y_{i-k}} = \emptyset \end{cases} \end{aligned} \tag{4.8}$$

where the function set $\mathcal{H}_{y_i|y_{i-k}}$ extends $\mathcal{H}_{y_i}$:

$$\mathcal{H}_{y_i|y_{i-k}} = \{f_{y_i} | R_S(f_{y_i}) < \delta \ \wedge \ R_S(f_{y_{i-k}}) < \delta'\} \tag{4.9}$$

and the function set $\mathcal{F}_{y_i|y_{i-k}}$ extends $\mathcal{F}_{y_i}$:

$$\mathcal{F}_{y_i|y_{i-k}} = \{f_{y_i} | R(f_{y_i}) < \delta \wedge R_S(f_{y_i}) < \delta \wedge R_S(f_{y_{i-k}}) < \delta'\} \tag{4.10}$$

Note that we use a different threshold δ' for $R_S(f_{y_{i-k}})$ in order to account for the difference between loss functions l_{i-k} and l_i. We do not require the true risk of intermediate concept $R(y_{i-k})$ to be lower than δ' because the objective is to analyze the achievable generalization *w.r.t.* predicting y_i.

4.3.3.2 Improved Generalization through Deep Supervision

A machine learning model for y_i suffers from overfitting when the solution f_{y_i} achieves low empirical risk $R_S(f_{y_i})$ over S but high true risk $R(f_{y_i})$. In other words, the higher the probability P_{y_i} is, the lower the chance that the trained model f_{y_i} overfits S is. One general strategy to reduce the overfitting is to increase the diversity and size of training set S. As such, the denominator $\mu(\mathcal{H}_{y_i})$ of Equation (4.7) decreases because fewer functions achieve low loss on more diverse data in general. In the following, we show that supervising intermediate concept y_{i-k} of y_i at some hidden layer is capable of removing

some incorrect solutions in $\mathcal{H}_{y_i} \setminus \mathcal{F}_{y_i}$ and thus improves the generalization because $\mathrm{P}_{y_i|y_{i-k}} \geq \mathrm{P}_{y_i}$.

First of all, we specify two assumptions which our analysis is based on. We first assume that y_{i-k} is a strict necessary condition of y_i (i.e. $\epsilon = 0$). Thus, there exists a deterministic function T which satisfies $y_{i-k} = T(y_i)$. Subsequently, we assume that if y_i' is a "good" prediction of y_i, then $T(y_i')$ is also a "good" prediction of its intermediate concept y_{i-k}. Formally, we assume:

$$\forall y_i, y_i' \in Q_i : \;\; l_i(y_i, y_i') \leq \delta \Rightarrow l_{i-k}(T(y_i), T(y_i')) \leq \delta' \tag{4.11}$$

where Q_i is the value space of concept y_i. In practice, we can find many tasks and their intermediate concepts satisfy assumption 4.11 when we use common loss functions and $\delta = \delta'$. We discuss more details on the above two assumptions in Section 4.3.3.3.

Given an intermediate concept y_{i-k} which satisfies both assumptions above, we now discuss how d_{i-k} (i.e. the supervision depth of y_{i-k}) affects the generalization capability of y_i in terms of $\mathrm{P}_{y_i|y_{i-k}}$ in the following propositions. Intuitively, supervising the intermediate concepts in the wrong order has no effect in improving the generalization.

Proposition 1. *If $d_{i-k} \geq d_i$, the generalization performance of y_i is not improved:*

$$\forall d_{i-k} \geq d_i, \quad \mathrm{P}_{y_i|y_{i-k}} = \mathrm{P}_{y_i} \tag{4.12}$$

Proof. We first consider the case when y_i and y_{i-k} both supervise the same

hidden layer: $d_i = d_{i-k}$. Given a sample set $(x, y_{i-k}, y_i) \sim \mathcal{D}$ and a function f_{y_i} which correctly predicts y_i for x: $y_i = f_{y_i}(x)$, we can construct $f_{y_{i-k}} = T \circ f_{y_i}$ to yield the correct prediction for y_{i-k}. As we know, a multi-layer perceptron (i.e. fully connected layers) with enough hidden units is able to represent any mapping function T following the universal approximation theorem [166]. Therefore, to approximate $f_{y_{i-k}} = T \circ f_{y_i}$, we can append fully connected layers which implement T to g_{d_i}: $g_{d_{i-k}} = T \circ g_{d_i}$. Based on the assumption of (4.11), for any function f_{y_i} in $\mathcal{F}_{y_i}$, there exists a corresponding function $f_{y_{i-k}} = T \circ f_{y_i}$ which satisfies $R_S(f_{y_{i-k}}) \leq \delta'$. This indicates that $\mathcal{H}_{y_i|y_{i-k}} = \mathcal{H}_{y_i}$ which in turn implies $\mathcal{F}_{y_i|y_{i-k}} = \mathcal{F}_{y_i}$. When $d_{i-k} > d_i$, hidden layers from d_i to d_{i-k} can be implemented to achieve an identity mapping and then follow the same analysis for the case $d_i = d_{i-k}$. As a consequence, Proposition 1 holds. □

Proposition 2. *There exists a d_{i-k} such that $d_{i-k} < d_i$ and the generalization performance of y_i is improved:*

$$\exists d_{i-k} < d_i, \quad \mathrm{P}_{y_i|y_{i-k}} \geq \mathrm{P}_{y_i} \tag{4.13}$$

Proof. From Equation 4.5 and 4.9, we observe that $\mathcal{H}_{y_i|y_{i-k}} \subset \mathcal{H}_{y_i}$ and $\mu(\mathcal{H}_{y_i|y_{i-k}}) < \mu(\mathcal{H}_{y_{i-k}})$. Thus, we obtain:

$$\mu(\mathcal{H}_{y_i|y_{i-k}}) \leq \min(\mu(\mathcal{H}_{y_i}), \mu(\mathcal{H}_{y_{i-k}})) \tag{4.14}$$

In addition, recall that a solution of y_i is assumed to satisfy its intermediate concept y_{i-k}. This suggests that there exists one or multiple d_{i-k}'s such that

the first d_{i-k} layers of each solution $f_{y_i} \in \mathcal{F}_{y_i}$ are contained in $f_{y_{i-k}} \in \mathcal{F}_{y_{i-k}}$. In other words, we can find a supervision depth d_{i-k} for y_{i-k} which satisfies:

$$\exists\, d_{i-k} < d_i, \quad \mu(\mathcal{F}_{y_i}) = \mu(\mathcal{F}_{y_i|y_{i-k}}) \tag{4.15}$$

As a result, Proposition 2 is proved by Equation 4.14 and Equation 4.15. The equality of Equation 4.13 is achieved when there exists a function $f_{y_{i-k}}$ in $\mathcal{F}_{y_{i-k}}$ for each $f_{y_i} \in \mathcal{F}_{y_i}$ such that f_{y_i} and $f_{y_{i-k}}$ share the first d_{i-k} hidden layers. However, as the toy example earlier, the hidden layers of some network solutions for y_i yield incorrect prediction of the intermediate concept y_{i-k}. This implies that $\mu(\mathcal{H}_{y_i|y_{i-k}}) \ll \min(\mu(\mathcal{H}_{y_i}), \mu(\mathcal{H}_{y_{i-k}}))$ in practice. □

To this end, we can improve the generalization of y_i via y_{i-k} by inserting the supervision of y_{i-k} before y_i. As a consequence, given a concept hierarchy $\mathcal{Y}_0 = (y_1, \cdots, y_m)$, the supervision depths of concepts $\{d_1, \cdots, d_m\}$ should be monotonically increasing: $1 \leq d_1 < \cdots < d_m$. We then extend Equation 4.14 to incorporate all available intermediate concepts with $\epsilon = 0$ of y_m:

$$\mu(\mathcal{H}_{y_m|y_{m-1},\cdots,y_1}) \leq \min_{y_i} \mu(\mathcal{H}_{y_i}) \;\; s.t. \;\; \forall i < j, d_i < d_j \tag{4.16}$$

As we report in Section 4.4.3, the empirical evidence shows that more intermediate concepts often greatly improves the generalization performance of the main task, which implies a large gap between two sides of Equation 4.16. Similar to Equation 4.15, we still have:

$$\exists\, d_1 < \cdots < d_m, \quad \mu(\mathcal{F}_{y_m}) = \mu(\mathcal{F}_{y_m|y_{m-1},\cdots,y_1}) \tag{4.17}$$

As a consequence, the generalization performance of y_m given its necessary

conditions $y_1, \cdots, y_{m-1}$ can be improved if we supervise each of them at appropriate depths $d_1, \cdots, d_{m-1}$ where $d_1 < \cdots < d_{m-1} < d_m$:

$$\exists d_1 < \cdots < d_m, \quad \mathrm{P}_{y_m|y_{m-1},\cdots,y_1} \geq \mathrm{P}_{y_m} \tag{4.18}$$

Furthermore, $\mathrm{P}_{y_m|y_{m-1},\cdots,y_1}$ is monotonically decreasing by removing intermediate concepts: $\mathrm{P}_{y_m|y_{m-1},\cdots,y_1} \geq \mathrm{P}_{y_m|y_{m-2},\cdots,y_1} \geq \cdots \geq \mathrm{P}_{y_m|y_1} \geq \mathrm{P}_{y_m}$. The more concepts applied, the better chance that the generalization is improved. In conclusion, deep supervision with intermediate concepts regularizes the network training by decreasing the number of incorrect solutions which generalize poorly to the test set.

4.3.3.3 Discussion

The role of ϵ. One of the assumptions for the analysis above is that y_{i-k} is a strict necessary condition with $\epsilon = 0$ for y_i. The intuition is that the strict necessary condition is "necessary" in the sense that it provides accurate information of intermediate states in an inference sequence. When $\epsilon > 0$, there is no guarantee to find a deterministic function T for y_i to map to y_{i-k}. Thus, y_{i-k} has the probability ϵ of being a different value from an expected intermediate state (i.e. $T(y_i)$). As a result, the monotonically increasing supervision order indicated by Equation 4.18 is no longer ensured. However, the architecture design suggested by our generalization analysis in Section 4.3.3.2 achieves the best performance in our empirical studies in Section 4.4.3. We believe that the generalization analysis in Section 4.3.3.2 is a good approximation for case with small ϵ in real applications. We leave the analytic quantification of how ϵ affects deep supervision to future work.

Understanding Assumption 4.11. Additionally, if Assumption 4.11 does not hold, both the numerator and denominator in Equation 4.8 decrease by different amounts. As a consequence, we cannot obtain Proposition 1 for all cases. However, many commonly used loss functions satisfy this assumption when $\delta = \delta'$. One simple example is when l_i and l_{i-k} are indicator functions (i.e. $l_i(y, y') = \mathbf{1}(y = y')$) for all i [2]. As such, $l_i(y, y') = l_{i-k}(T(y), T(y'))$ when $\epsilon = 0$ and thus Assumption 4.11 is satisfied. Another example can be that l_i and l_{i-k} are both L2 loss (i.e. $l_i(y, y') = \|y = y'\|^2$) and T is a projection function where $T(y) = Py$ and P is a projection (i.e. $P^2 = P$). In this case, $l_i(y, y') = \|y - y'\|^2 \geq \|P(y - y')\|^2 = l_{i-k}(T(y), T(y'))$.

The risk threshold δ is not constrained to any particular value range. Although we care more about small δ rather than the large one in practice, the results of our generalization analysis can be applied for arbitrary δ. How to train a deep model to obtain the empirical risk lower than δ is beyond the scope of this work.

DSN as a special case. Because a task is also a necessary condition of itself, our deep supervision framework actually contains DSN[35] as a special case where each intermediate concept y_i is the main task itself. A potential limitation of DSN [35] predicted by our analysis is that it may over-regularize a deep model and decrease its learning power. To illustrate this, we set the first intermediate concept $y_1 = y_m$. Thus, the first d_1 hidden layers are forced to directly predict y_m. Each $f_{d_1} \in \mathcal{F}_{y_1}$ can be trivially applied to construct $f_{d_m} \in \mathcal{F}_{y_m}$ by forcing an identity function for layers from d_1 to d_m.

[2] Note that the indicator function can be applied to discrete and continuous values of y and y'.

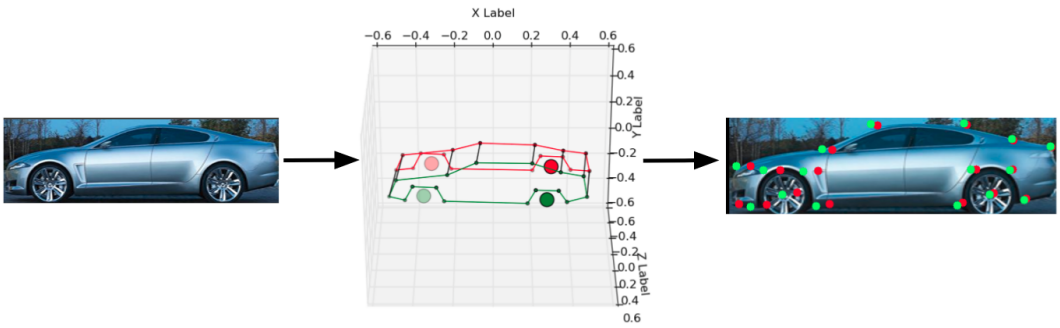


Figure 4.3: The objective of 2D/3D keypoint localization is to estimate 3D shape structure (middle) and 2D keypoint projection (right) from a single color image (left). A 3D shape structure contains keypoints as its joints and their inter-connection as its skeleton.

This suggests that $\mathcal{F}_{y_m}$ is mainly constrained by $\mathcal{F}_{y_1}$. Therefore, the learning capacity of an N-layer CNN becomes highly dependent on its first d_1 layers, which in turn makes a deep model shallower.

4.4 Keypoint Localization

In this section, we show the first instantiation of our deep supervision methodology described in Section 4.3. The objective is to predict 2D and 3D object skeletons [167] given a single test image, as shown in Figure 4.3. Our approach is in the spirit of [152, 151] which exploit object pose as an auxiliary shape concept to aid shape interpretation and mental rotation. We combine this early intuition with the discriminative power of modern CNNs by deeply supervising for multiple shape concepts such as object pose. As such, deep supervision teaches the CNN to sequentially model intermediate goals to parse 2D/3D object skeletons across large intra-class appearance variations

and occlusion.

We use 3D skeletons [167] as our 3D shape representation, where semantically meaningful object parts (such as the wheels of a car) are represented by 3D keypoints and their connections define 3D topology of an object category. This representation is more efficient than 3D volumes [79]ïijN̆ meshes [76, 78, 77, 74, 80] and 3D bounding boxes [168] in conveying the semantic information necessary for shape reasoning in various applications.

Few of existing datasets provide annotations of intermediate concepts and even the 3D skeleton for real images, which makes the training of CNN infeasible in practice. To solve the scarcity of 3D annotation, we render 3D CAD models to generate large-scale synthetic datasets to be used for training, as described in Section 4.4.1. In addition, we simulate challenging occlusion configurations between objects to enable robust data-driven occlusion reasoning (in contrast to earlier model-driven attempts [169, 72]).

We impose deep supervision of intermediate shape concepts, such as object pose and part visibility, within the hidden layers of a CNN. We find deep supervision to be the critical element to bridge the synthetic and real world. Figure 4.4 introduces our framework and Figure 4.9 illustrates an instance of a CNN deeply supervised by intermediate shape concepts for 2D/3D keypoint localization. We denote our network as "DISCO" short for Deep supervision with Intermediate Shape COncepts.

Most existing approaches [78, 74, 170, 71, 72] estimate 3D geometry by comparing projections of parameterized shape models with separately predicted 2D patterns, such as keypoint locations or heat maps. This makes prior

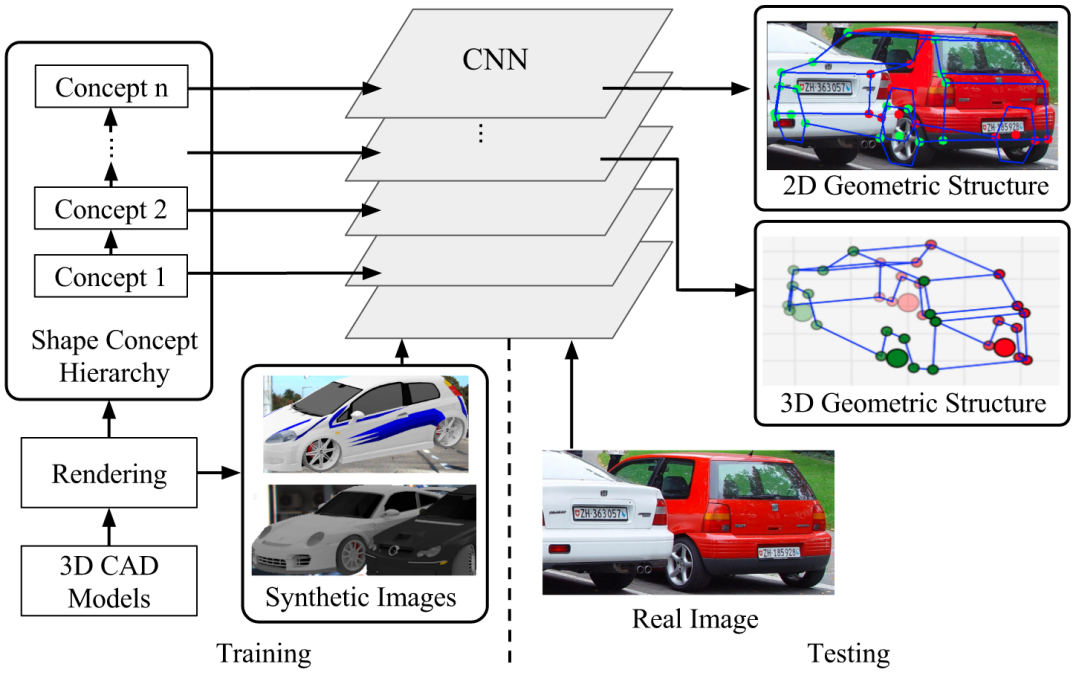


Figure 4.4: Overview of our approach. We use synthetic training images with intermediate shape concepts to deeply supervise the hidden layers of a CNN. At test time, given a single real image of an object, we demonstrate accurate localization of semantic parts in 2D and 3D, while being robust to intra-class appearance variations and occlusions.

methods sensitive to partial view ambiguity [171] and incorrect 2D structure prediction. Moreover, scarce 3D annotation of real image further limits their performance. In contrast, our method is trained on synthetic data only and generalizes well to real images. We find deep supervision to be the critical element to bridge the synthetic and real world. In particular, our deep supervision scheme empirically outperforms the single-task architecture, and multi-task networks which supervise all the concepts at the final layer. Further, we quantitatively demonstrate significant improvements over prior state-of-the-art for 2D/3D keypoint prediction on PASCAL VOC, PASCAL3D+[32],

IKEA[33] and an extension on the KITTI[172] dataset (KITTI-3D). These observations confirm that intermediate concepts regularize the learning of 3D shape in the absence of photorealism such as material and illumination in rendered training data.

In summary, we make the following contributions in this work:

- We show the utility of rendered data with access to intermediate shape concepts. We model occlusions by rendering multiple object configurations, which presents a novel route to exploiting 3D CAD data for parsing cluttered scenes.
- We present a CNN architecture where its hidden layers are supervised by a sequence of intermediate shape concepts for the main task of 2D and 3D object geometry estimation
- Our approach exhibits markedly improved generalization from synthetic to real images compared to standard end-to-end training. We empirically demonstrate state-of-the-art performance on 2D/3D semantic part localization and object classification on several public benchmarks. In some experiments, the proposed approach even outperforms the state-of-the-art methods trained on real images. We also demonstrate superior performance to baselines including the conventional multi-task supervision and different orders of intermediate concepts.

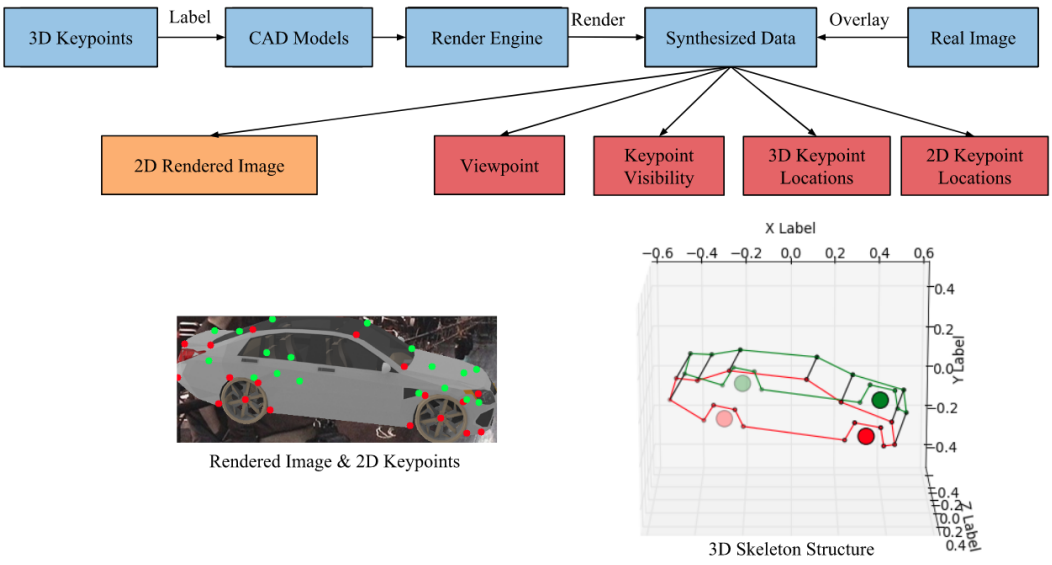


Figure 4.5: Visualization of our rendering pipeline (top-left), DISCO network (bottom-left), an example of rendered image and its annotations of 2D keypoints (top-right) as well as 3D skeleton (bottom-right).

4.4.1 Synthetic Training Data

Our approach needs a large amount of training data because it is based on deep CNNs. It also requests finer grained labels than many visual tasks such as object detection. Furthermore, we aim for the method to work for heavily cluttered scenes. Therefore, we generate synthetic images that simulate realistic occlusion configurations involving multiple objects in close proximity. Specifically, we render multiple objects simulating occlusion patterns arising in scenes due to multiple objects apart from truncations. To our knowledge, rendering cluttered scenes that comprise of multiple CAD models is a novelty of our approach, although earlier work [93, 81] used real image cut-outs for bounding box level localization.

An overview of the rendering process is shown in the Figure 4.5. We pick

Figure 4.6: Visualization of the diversity of synthetic cars. We vary the scale, resolution, illumination, viewpoint and real image background to create large appearance variation.

a small subset of CAD models from ShapeNet [173] for a given object category and manually annotate 3D keypoints on each CAD model. We use meshlab for 3D keypoint annotation. Note that the 3D keypoint groundtruth is normalized in scale because CAD models in ShapeNet v0 are normalized cars as well. Next, we render each CAD model via some rendering engine with randomly sampled graphics parameters including camera viewpoint, number/strength of light sources, and surface gloss reflection. In our implementation, we use Blender for the ease of distributed computation and good quality of rendering by back-tracing. We visualize the diversity of an example rendered car in the Figure 4.6. Finally, we follow [65] to overlay the rendered images on real backgrounds to avoid over-fitting. We crop the object from each rendered image and extract the object viewpoint, 2D/3D keypoint locations and their visibility states from Blender as the training labels. We use KITTI [172] as the source of real image background for car and SUN [174] for furniture objects. In the bottom of Figure 4.5, we show an example of rendering and its 2D/3D annotations.

Figure 4.7: Examples of synthesized training images for simulating the multi-car occlusion.

To model multi-object occlusion, we randomly select two different object instances and place them onto the groundplane while being close to each other without overlapping in 3D space. During rendering, we compute the occlusion ratio of each instance by calculating the fraction of visible 2D area versus the complete 2D projection of CAD model. Keypoint visibility is computed by ray-tracing. We select instances with occlusion ratios ranging from 0.4 to 0.9. Figure 4.7 shows two training examples where cars are occluded by other nearby cars. For truncation, we randomly select two image boundaries (left, right, top, or bottom) of the object and shift them by $[0, 0.3]$ of the image size along that dimension.

Last, in Figure 4.8, we qualitatively compare our simulated data with the real data on the challenging cases including truncation and car-to-car occlusion. We can see that our simulated data is able to capture the scene layout under each specific type of occlusion, although the photorealism of synthetic data can be further improved.

4.4.2 Network Architecture

We define a 3D skeleton for each object class where joints or keypoints represent semantic parts, and their connections define 3D object geometry. Given a

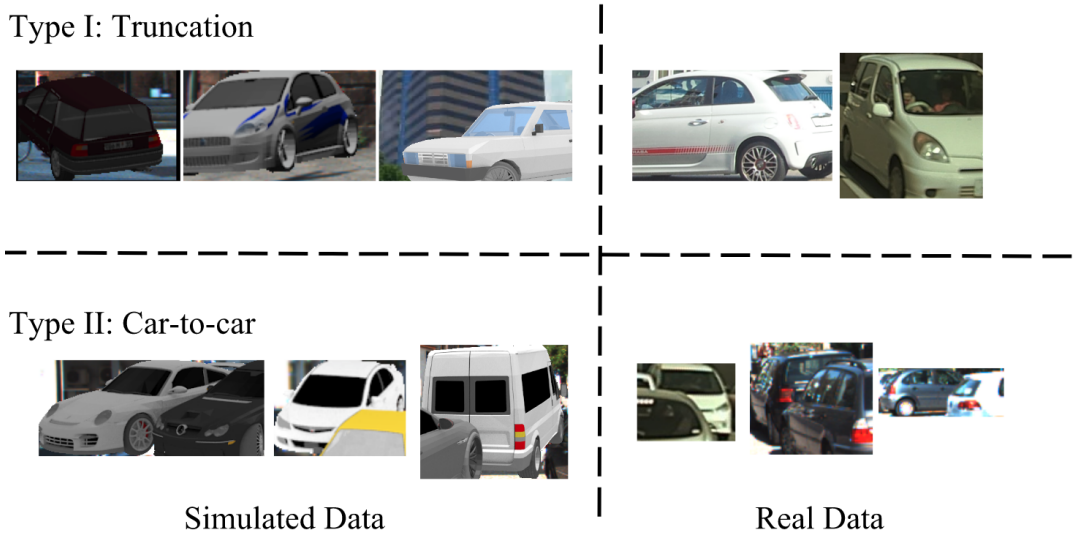


Figure 4.8: Examples of simulated truncated data (top-left), truncated real data (top-right), simulated car-to-car occlusion data (bottom-left) and real car-to-car occlusion data (bottom-right).

single real RGB image of an object, our goal is to predict the keypoint locations in image coordinates as well as normalized 3D coordinates while inferring their visibility states. X and Y coordinates of 2D keypoint locations are normalized to $[0, 1]$ along the image width and height, respectively. 3D keypoint coordinates are centered at origin and scaled to set the longest dimension along X,Y,Z to unit length. Note that 2D/3D keypoint locations and their visibility all depend on the specific object pose with respect to the camera viewpoint.

To set up the concept hierarchy for 2D/3D keypoint localization, we have chosen in order, object orientation y_1, which is needed to predict keypoint visibility y_2, which roughly depicts the 3D structure prediction y_3, which finally leads to 2D keypoint locations y_4 including ones that are not visible

in the current viewpoint. We impose the supervision of the concept hierarchy $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$ into a CNN as shown in Figure 4.9 and minimize Equation 4.2 to compute the network parameters.

We emphasize that the above $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$ is not a 0-error concept hierarchy because object pose (y_1), and 3D keypoint location (y_3) are not strict necessary conditions for visibility (y_2), and 2D keypoint location (y_4), respectively. However, we posit that the corresponding residuals (ϵ's) of $\mathcal{Y}$ are small. First, knowing object pose constrains keypoint visibilities to such an extent, that prior work has chosen to use ensembles of 2D templates for visual object parsing [175, 93]. Second, there is a long and fruitful tradition in computer vision, starting from Marr's seminal ideas [152] to leverage 3D object representations as a tool for 2D recognition. If we know the exact shape model and object pose, the ensemble of 2D keypoint locations determines 3D keypoint locations via back-projection up to scale. For instance, one may jointly optimize the shape model and object pose to fit its projections on 2D keypoints [176], which computes the 3D keypoint locations with small error. For our own experiments, we use 2D keypoint annotations to generate 3D (pseudo-)groundtruth for real images, which is shown in Section 4.4.3.1. In sum, our present choice of $\mathcal{Y}$ is an approximate realization of a 0-error concept hierarchy which nonetheless draws inspiration from our analysis, and works well in practice.

Our network resembles the VGG network [43] and consists of deeply stacked 3×3 convolutional layers. Unlike VGG, we remove local spatial pooling between convolutional layers. This is motivated by the intuition that

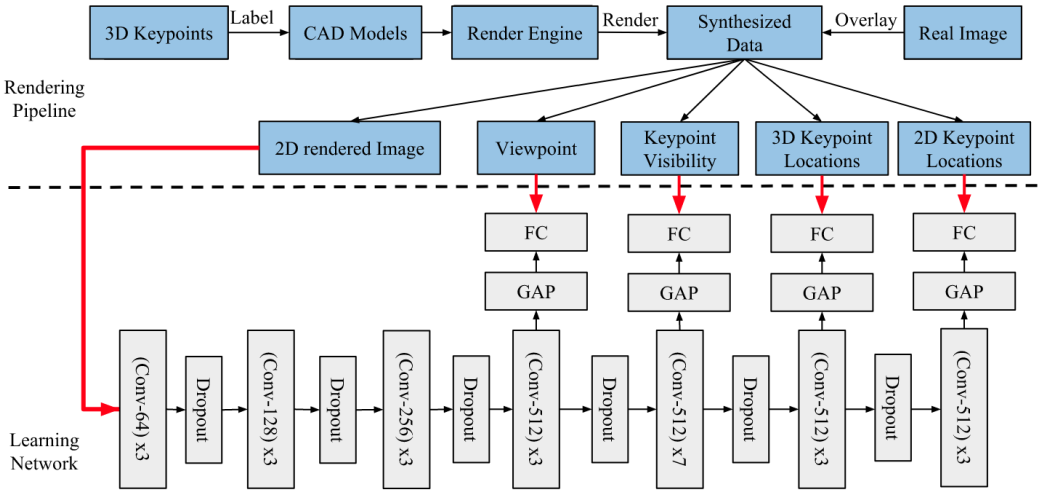


Figure 4.9: Visualization of our rendering pipeline (top-left), DISCO network (bottom-left), an example of rendered image and its annotations of 2D keypoints (top-right) as well as 3D skeleton (bottom-right).

spatial pooling leads to the loss of spatial information. Further, we couple each convolutional layer with batch normalization [44] and ReLU, which defines $h_{d_i}(x, W_{d_i})$. The output layer $g_{d_i}(\cdot, V_{d_i})$ at depth d_i for task y_i is constructed with one global average pooling (GAP) layer followed by one fully connected (FC) layer with 512 neurons, which is different from stacked FC layers in VGG. The GAP layer averages filter responses over all spatial locations within the feature map. In Section 4.4.3.1, we empirically show that these two changes are critical to significantly improve the performance of VGG-like networks for 2D/3D landmark localization.

We follow the common practice of employing dropout [9] layers between the convolutional layers, as an additional means of regularization. At layers 4,8,12, we perform the downsampling using convolution layers with stride 2. Our network design shares the same spirit to Resnet [45] but no spatial pooling

and residual connection is used. We empirically observe that the dropout layer leads to a better performance than the residual connection for 2D/3D landmark localization. The bottom-left of Figure 4.9 illustrates the details of our network architecture. "(Conv-A)xB" means A stacked convolutional layers with filters of size BxB. We deploy 25 convolutional layers in total.

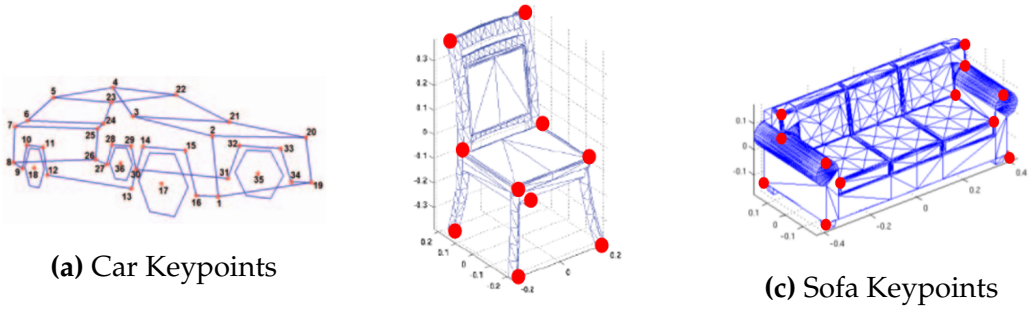
We use L2 loss at all points of supervision. In practice, we only consider the azimuth angle of the object viewpoint with respect to a canonical pose. We further discretize the azimuth angle into K bins and regress it to a one-hot encoding (the entry corresponding to the predicted discretized pose is set to 1 and all others to 0). Keypoint visibility is also represented by a binary vector with 1 indicating occluded state of a keypoint. During training, each loss is backpropagated to train the network jointly.

4.4.3 Experiment

We empirically demonstrate competitive or superior performance for 2D/3D keypoint localization over several state-of-the-art methods, on multiple datasets: KITTI-3D (Section 4.4.3.1), PASCAL VOC (Section 4.4.3.2), PASCAL3D+ [32] (Section 4.4.3.3) and IKEA [33] (Section 4.4.3.4).

CAD Model.

For data synthesis, we sample CAD models of 472 cars, 100 sofas, 100 chairs and 62 beds from ShapeNet [173]. We follow Zia et al. [72] to annotate 36 keypoints for each car CAD model. The 3D skeleton of a car is shown in Figure 4.10a. Each keypoint represents a particular semantic part of the car such as the wheel and corners of the front/back windshield. For IKEA



(a) Car Keypoints

(b) Chair Keypoints

(c) Sofa Keypoints

Figure 4.10: Visualization of keypoint definitions on car, chair and sofa classes. Invisible keypoints are not shown in Figure 4.10c.

dataset, we annotate 14 keypoints for both chair and sofa CAD models as shown in Figure 4.10b and Figure 4.10c, respectively. Note that the definition of keypoints on the sofa seating area (shown in Figure 4.10c) are inconsistent with 3D-INN[73]. Additionally, the keypoints on armrests of a chair are merged to the keypoints on the seating area if armrests do not exist. We synthesize 600k car images including occluded instances and 300k images of fully visible furniture (chair+sofa+bed). We pick rendered images of 5 CAD models from each object category as validation set.

KITTI-3D Annotation. We introduce KITTI-3D with annotations of 3D keypoint and occlusion type on 2040 car images from [172]. We label car images with one of four occlusion types: no occlusion (or fully visible cars), truncation, multi-car occlusion (target car is occluded by other cars) and occlusion cause by other objects. The number of images for each type is 788, 436, 696 and 120, respectively.

To obtain 3D groundtruth for these car images, we fit a PCA model trained on 3D keypoint annotation on CAD data, by minimizing the 2D projection

error for known 2D landmarks provided by Zia et al. [72] and object pose from KITTI [172]. First, we compute the mean shape M and 5 principal components $P_1, \cdots, P_5$ from 3D skeletons of our annotated CAD models. M and P_i ($1 \leq i \leq 5$) are 3×36 matrices where each column contains 3D coordinates of a keypoint. Thus, the 3D object structure X is represented as $X = M + \sum_{i=1}^{5} \alpha_i P_i$, where α_i is the weight for P_i. To avoid distorted shapes caused by large α_i, we constrain α_i to lie within $-2.7\sigma_i \leq \alpha_i \leq 2.7\sigma_i$ where σ_i is the standard deviation along the i^{th} principal component direction. Next, given the groundtruth pose T, we compute 3D structure coefficients $\alpha = \{\alpha_i\}$ that minimize the projection error with respect to 2D ground truth Y:

$$\begin{aligned} \alpha^* &= \arg_\alpha \min_K \|\Pr(KT(M + \sum_{i=1}^{N} \alpha_i P_i) - Y\|_2^2 \\ &= \arg_\alpha \min_{s,\beta} \|s\Pr(T(M + \sum_{i=1}^{N} \alpha_i P_i)) + \beta - Y\|_2^2 \\ &s.t. \ -2.7\sigma_i \leq \alpha_i \leq 2.7\sigma_i \end{aligned} \tag{4.19}$$

where the camera intrinsic matrix is $K = [s_x, 0, \beta_x; 0, s_y, \beta_y; 0, 0, 1]$ with the scaling $s = [s_x; s_y]$ and shifting $\beta = [\beta_x; \beta_y]$. $\Pr(x)$ computes the 2D image coordinate from 2D homogeneous coordinate x. In practice, to obtain the ground truth with even higher quality, we densely sample object poses $\{T_j\}$ in the neighborhood of T and solve (4.19) by optimizing $\{\alpha_i\}, \beta, s$ given a fixed T_j and then search for the lowest error among all sampled T_j. We only provide 3D keypoint labels for fully visible cars because we do not have enough visible 2D keypoints for most of the occluded or truncated cars and thus obtain rather crude 3D estimates for such cases. Figure 4.11 shows some examples of 3D

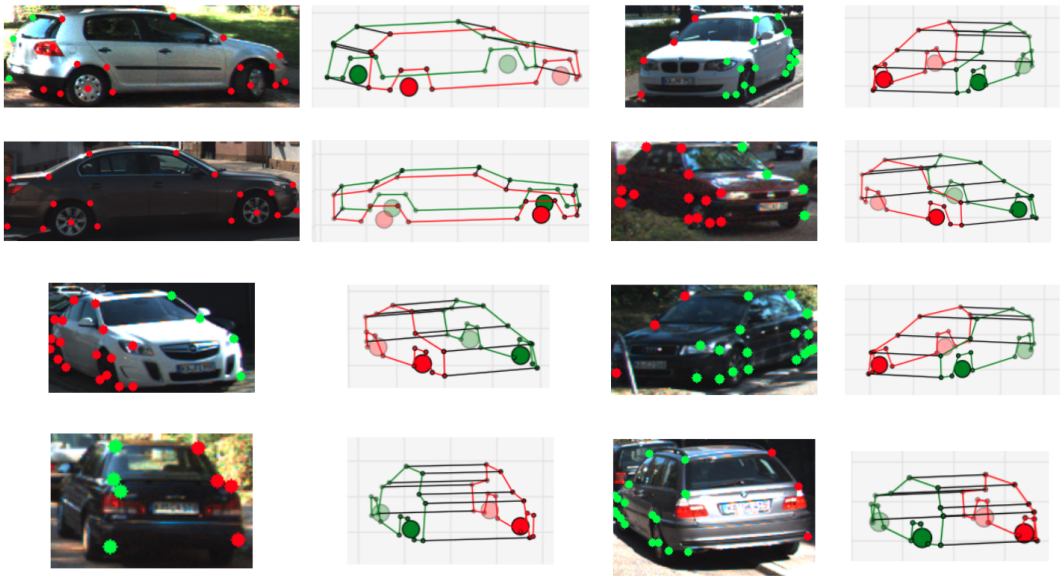


Figure 4.11: Examples of 2D and 3D annotations in KITTI-3D. Visible 2D keypoints annotated by Zia et al.[72] are shown on the car images. The corresponding 3D skeleton is shown to the right of each image.

groundtruth computed for KITTI-3D dataset.

Evaluation metric. We use PCK and APK metrics [31] to evaluate the performance of 2D keypoint localization. A 2D keypoint prediction is correct when it lies within the radius $\alpha * L$ of the ground truth, where L is the maximum of image height and width and $0 < \alpha < 1$. PCK is the percentage of correct keypoint predictions given the object location and keypoint visibility. APK is the mean average precision of keypoint detection computed by associating each estimated keypoint with a confidence score. In our experiments, we use the regressed values of keypoint visibility as confidence scores. We extend 2D PCK and APK metrics to 3D by defining a correct 3D keypoint prediction whose euclidean distance to the ground truth is less than α in normalized coordinates.

Training details. We set loss weights of visibility, 3D and 2D keypoint locations $\{\lambda_i\}$ to 1 and object pose to 0.1. We use stochastic gradient descent with momentum 0.9 to train the proposed CNN from scratch. Our learning rate starts at 0.01 and decreases by one-tenth when the validation error reaches a plateau. We set the weight decay to 0.0001, resize all input images to 64x64 and use batch size of 100. We initialize all weights using Glorot and Bengio [177]. For car model training, we form each batch using a mixture of fully visible, truncated and occluded cars, numbering 50, 20 and 30, respectively. For the furniture, each batch consists of 70 fully visible and 30 truncated objects randomly sampled from the joint synthetic image set of chair, sofa and bed.

4.4.3.1 KITTI-3D

We compare our method with DDN [87] and WarpNet [88] for 2D keypoint localization and Zia et al. [72] for 3D structure prediction. We use the original source codes for these methods. However, we enhance WarpNet (denoted as WN-gt-yaw) to leverage groundtruth poses of test images. Specifically, we retrieve 30 labeled synthetic car images with the same pose for landmark localization using the same CNN architecture proposed in [88], and then compute the median of predicted landmark locations as the final result. Additionally, we perform an ablative analysis of DISCO. First, we replace all intermediate supervisions with the final labels, as DSN [35] does, for 2D (DSN-2D) and 3D (DSN-3D) structure prediction. Next, we incrementally remove the deep supervision used in DISCO one by one. DISCO-vis-3D-2D, DISCO-3D-2D, plain-3D, and plain-2D are networks without pose, pose+visibility, pose+visibility+2D

Method	2D Keypoint				
	Full	Truncation	Multi-Car Occ	Other Occ	All
DDN [87]	67.6	27.2	40.7	45.0	45.1
WN-gt-yaw* [88]	88.0	76.0	81.0	82.7	82.0
Zia et al. [72]	73.6	NA			
DSN-2D	45.2	48.4	31.7	24.8	37.5
plain-2D	88.4	62.6	72.4	71.3	73.7
plain-all	90.8	72.6	78.9	80.2	80.6
DISCO-3D-2D	90.1	71.3	79.4	82.0	80.7
DISCO-vis-3D-2D	92.3	75.7	81.0	83.4	83.4
DISCO-(3D-vis)	91.9	77.6	82.2	**86.1**	84.5
DISCO-reverse	30.4	29.7	22.8	19.6	25.6
DISCO-Vgg	83.5	59.4	70.1	63.1	69.0
DISCO	**93.1**	**78.5**	**82.9**	85.3	**85.0**
DISCO(Det)	95.9	78.9	87.7	90.5	88.3

Table 4.2: PCK[$\alpha = 0.1$] accuracies (%) of different methods for 2D keypoint localization on KITTI-3D dataset. WN-gt-yaw [88] uses groundtruth pose of the test car. The bold numbers indicates the best result on groundtruth object bounding boxes. The last row presents the accuracies of DISCO on detection results from RCNN[46].

and pose+visibility+3D, respectively. Further, we change the locations of the intermediate supervision signals. plain-all shifts supervision signals to the final convolutional layer. DISCO-(3D-vis) switches 3D and visibility in DISCO, and DISCO-reverse reverses the entire order of supervisions in DISCO. Finally, DISCO-VGG replaces stride-based downsampling and GAP in DISCO with non-overlapping spatial pooling (2x2) and a fully connected layer with 512 neurons, respectively. All methods are trained on the same set of synthetic training images and tested on real cropped cars on ground truth locations in KITTI-3D.

In Table 4.2 and Table 4.3, we report PCK accuracies for various methods[3]

[3]We cannot report Zia et al.[72] on occluded data because only a subset of images has valid result in those classes.

Method	3D	3D-yaw
Zia et al. [72]	73.5	7.3
DSN-3D	68.3	12.5
plain-3D	90.6	6.5
plain-all	92.9	3.9
DISCO-3D-2D	94.3	3.1
DISCO-vis-3D-2D	95.2	2.3
DISCO-(3D-vis)	94.2	2.3
DISCO-reverse	54.8	13.0
DISCO-Vgg	89.7	6.8
DISCO	**95.3**	**2.2**
DISCO(Det)	95.5	2.1

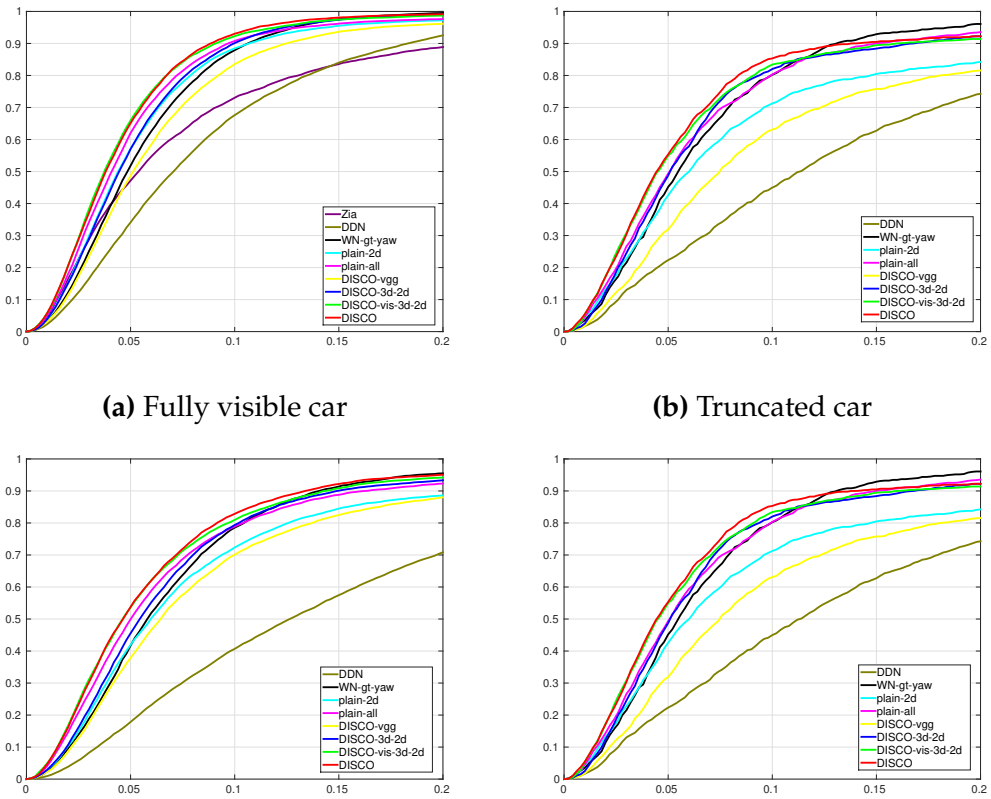
Table 4.3: PCK[$\alpha = 0.1$] accuracies (%) of different methods for 3D keypoint localization on KITTI-3D dataset. Last column represents angular error in degrees. WN-gt-yaw [88] uses groundtruth pose of the test car. The bold numbers indicates the best result on groundtruth object bounding boxes. The last row presents the accuracies of DISCO on detection results from RCNN [46].

and the mean error of estimated yaw angles "3D-yaw" over all fully visible cars. Additionally, Figure 4.12 and Figure 4.13 demonstrate 2D and 3D PCK curves of various methods, respectively. This object-centric yaw angle is defined as the angle of car head direction with respect to X axis on the XZ plane or ground plane. Given an estimated 3D skeleton, we first project all keypoints onto the XZ plane and compute the average direction over all lines perpendicular to the left-to-right correspondence lines. Then, we can obtain the yaw angle as the angle of this average direction with respect to the X axis. In practice, we use lines between keypoints (1,19),(17,35) and (5,23) as correspondence lines where keypoint id is shown in Figure 4.10a. Finally, we compute the absolute error between the estimated yaw direction and the ground truth yaw and average the error over all test images.

Training Data			Test Data		
Full	Truncation	Multi-Car Occlusion	Full	Truncation	Occlusion
✓			91.8	53.6	68.3
	✓		89.9	73.8	61.7
		✓	91.3	74.7	82.7
✓	✓		92.9	71.3	63.4
✓		✓	92.5	73.2	**84.1**
	✓	✓	90.5	70.4	81.2
✓	✓	✓	**93.1**	**78.5**	83.2

Table 4.4: Ablative study of different training data sources. PCK[$\alpha = 0.1$] accuracies (%) of DISCO for 2D keypoint localization on KITTI-3D dataset.

We observe that DISCO outperforms competitors in both 2D and 3D keypoint localization across all occlusion types. Moreover, we observe a monotonic increase in 2D and 3D accuracy with increasing supervision: plain-2D or plain-3D $<$ DISCO-3D-2D $<$ DISCO-vis-3D-2D $<$ DISCO. Further, plain-all is superior to plain-2d and plain-3d, while DISCO exceeds plain-all by 4.4% on 2D-All and 2.4% on 3D-Full. These experiments confirm that joint modeling of 3D shape concepts is better than independent modeling. We attribute this success to the complementary nature of our labels and the regularization effect via deep supervision. Moreover, alternative supervision orders (DISCO-reverse, DISCO-(3D-vis)) are found to be inferior to the proposed order which captures underlying structure between shape concepts. Last, DISCO-VGG performs significantly worse than DISCO by 16.0% on 2D-All and 5.6% on 3D-Full, which validates our removal of local spatial pooling and adopt global average pooling. In conclusion, the proposed deep supervision architecture coupled with intermediate shape concepts improves the generalization ability of CNN. As more concepts are introduced in the "correct" order, we observe



(a) Fully visible car

(b) Truncated car

(c) Multi-car occlusion

(d) Other occlusion

Figure 4.12: 2D PCK curves of comparative methods, variants of DISCO and DISCO on fully visible car (Figure 4.12a), truncated car (Figure 4.12b), multi-car occlusion (Figure 4.12c) and other occlusion (Figure 4.12d. In each figure, X axis stands for α of PCK and Y axis represents the accuracy.

improvement in performance.

We also conduct an ablative study of training data with different occlusion types. Table 4.4 demonstrates 2D keypoint localization accuracies over different occlusion categories on KITTI-3D given various combination of training data. "Occ." stands for test examples with multi-object occlusions where the occluder is either another car or a different object such as a pedestrian. As we can see, DISCO trained on fully visible cars alone achieves much worse

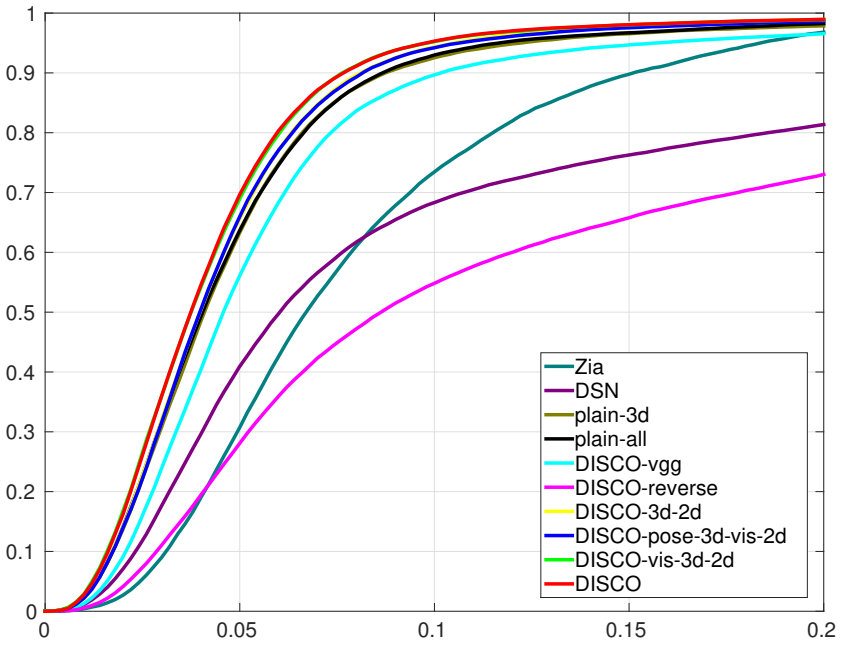


Figure 4.13: 3D PCK curves of different methods and variants of DISCO.

performance on truncated and occluded test data than when trained on data with simulated truncation and multi-car occlusion. We observe that multi-car occlusion data is also helpful in modeling truncation cases, and the network trained by multi-car data obtains the second best result on truncated cars. The best overall performance is obtained by including all three types of examples (no occlusion, multi-car occlusion, truncation), emphasizing the efficacy of our data generation strategy.

Finally, we evaluate DISCO on detection bounding boxes computed from RCNN[46] with IoU$>$ 0.7 to the groundtruth of KITTI-3D. "DISCO-Det" in the last row of Table 4.2 and Table 4.3 shows PCK accuracies of DISCO using detection results. The 2D/3D keypoint localization accuracies even exceeds the performance of DISCO using groundtruth bounding boxes by 3.3% on

PCK [$\alpha = 0.1$]	Full	Full [$\alpha = 0.2$]	Occluded	Big Image	Small Image	All [APK $\alpha = 0.1$]
Long[178]	55.7	NA				
VpsKps[86]	81.3	88.3	**62.8**	**90.0**	67.4	40.3
DSN-2D	75.4	87.8	54.5	85.5	63.3	NA
plain-2D	76.7	90.6	50.4	80.6	69.4	NA
plain-all	75.9	90.4	53.0	82.4	65.1	41.7
DISCO-reverse	64.5	84.5	41.2	55.5	67.0	24.9
DISCO-3D-2D	81.5	92.0	61.0	87.6	73.1	NA
DISCO	**81.8**	**93.4**	59.0	87.7	**74.3**	**45.4**

Table 4.5: PCK[$\alpha = 0.1$] accuracies (%) of different methods for 2D keypoint localization on the car category of PASCAL VOC. Bold numbers indicate the best results.

2D-All and 0.2% on 3D-All. We believe this can be attributed to the fact that 2D groundtruth locations in KITTI do not tightly bound the object areas because they are only the projections of 3D groundtruth bounding boxes. This result shows that DISCO is robust to imprecise 2D bounding boxes from detection algorithms like RCNN. In conclusion, DISCO can generalize the structure patterns learned from synthetic images to real data even with occlusion and significantly outperform other existing methods.

4.4.3.2 PASCAL VOC

We evaluate DISCO on the PASCAL VOC 2012 dataset for 2D keypoint localization [31]. Unlike KITTI-3D where car images are captured on real roads and mostly in low resolution, PASCAL VOC contains car images with larger appearance variations and heavy occlusions. In Table 4.5, we compare our results with the state-of-the-art [86, 178] on various sub-classes of the test

set: fully visible cars (denoted as "Full"), occluded cars, high-resolution (average size 420x240) and low-resolution images (average size 55x30). Please refer to [86] for details of the test setup. Note that these methods [86, 178] are trained on real images, whereas DISCO training exclusively leverages synthetic training data.

We observe that DISCO outperforms [86] by 0.6% and 5.1% on PCK at $\alpha = 0.1$ and $\alpha = 0.2$, respectively. In addition, DISCO is robust to low-resolution images, improving 6.9% accuracy on low-resolution set compared with [86]. This is critical in real perception scenarios where distant objects are small in images of street scenes. However, DISCO is inferior on the occluded car class and high-resolution images, attributable to our use of small images (64x64) for training and the fact that our occlusion simulation does not capture the complex occlusions in non-road scenes. Finally, we compute APK accuracy at $\alpha = 0.1$ for DISCO on the same detection candidates used in [86][4]. We can see that DISCO outperforms [86] by 5.1% on the entire car dataset (Full+Occluded). This suggests DISCO is more robust to noisy detection results and more accurate on keypoint visibility inference than [86]. We attribute this to global structure modeling of DISCO during training where the full set of 2D keypoints resolves the partial view ambiguity whereas traditional methods like [86] only are supervised with visible 2D keypoints.

Note that some definitions of our car keypoints [72] are slightly different from [31]. For example, we annotate the bottom corners of the front windshield whereas [31] labels the side mirrors. In our experiments, we ignore

[4]We run the source code [86] to obtain the same object candidates.

Method	CAD alignment GT	Manual GT
VDPM-16 [32]	NA	51.9
Xiang et al. [179]	64.4	64.3
Random CAD [32]	NA	61.8
GT CAD [32]	NA	67.3
DSN-2D	66.4	63.3
plain-2D	67.4	64.3
plain-all	66.8	64.2
DISCO-reverse	54.2	56.0
DISCO	**71.2**	**67.6**

Table 4.6: Object segmentation accuracies (%) of different methods on PASCAL3D+. Best results are shown in bold.

this annotation inconsistency and directly assess the prediction results. We reemphasize that unlike [178, 86], we do not use the PASCAL VOC train set. Thus, even better performance is expected when real images with consistent labels are used for training.

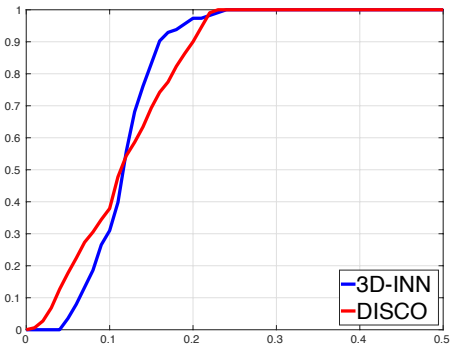
4.4.3.3 PASCAL 3D

PASCAL3D+ [32] provides object viewpoint annotations for PASCAL VOC objects by manually aligning 3D object CAD models onto the visible 2D keypoints. Because only a few CAD models are used for each category, the 3D keypoint locations are only approximate. Thus, we use the evaluation metric proposed by [32] which measures 2D overlap (IoU) against projected model mask. With a 3D skeleton of an object, we are able to create a coarse object mesh based on the geometry and compute segmentation masks by projecting coarse mesh surfaces onto the 2D image based on the estimated 2D keypoint locations. In detail, based on the 3D skeleton of the car category as shown in Figure 4.10a, we can define a rough 3D mesh using the 3D keypoints. For

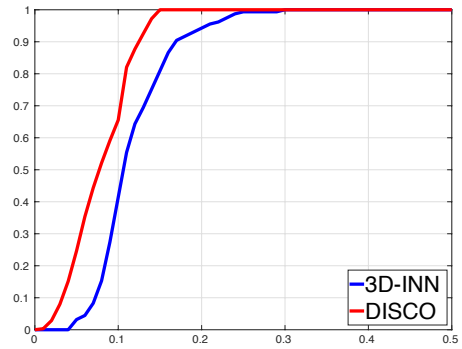
example, the rectangular area bounded by keypoint 1,2,20,19 forms the surface of the head of a car. Recall that our network DISCO localizes the complete set of 2D keypoints regardless of the visibility for an object. Therefore, we can compute the projected area of each predefined mesh surface and the union of all projected surfaces is the instance segmentation mask. Note that the surface of a car wheel is a hexagon where the center is the wheel keypoint and the corners are defined based on the surrounding keypoints.

Table 4.6 reports object segmentation accuracies on two types of ground truth. The column "Manual GT" uses manual pixel-level annotation provided by PASCAL VOC 2012, whereas "CAD alignment GT" uses 2D projections of aligned CAD models as ground truth. Note that "CAD alignment GT" covers the entire object extent in the image including regions occluded by other objects. DISCO significantly outperforms a state-of-the-art method [81] by 4.6% and 6.6% despite using only synthetic data for training. Moreover, on "Manual GT" benchmark, we compare DISCO with "Random CAD" and "GT CAD" which stand for the projected segmentation of randomly selected and ground truth CAD models respectively, given ground truth object pose. Note that "GT CAD" sets the upper bound performance[5] for methods that estimate shape based on 3D model retrieval and alignment. We find that DISCO yields even superior performance to "GT CAD". This provides evidence that joint modeling of 3D geometry manifold and viewpoint is better than the pipeline of object retrieval plus alignment. Finally, we note *two orders of magnitude faster inference* of a forward pass of DISCO during testing compared with sophisticated CAD alignment approaches [72].

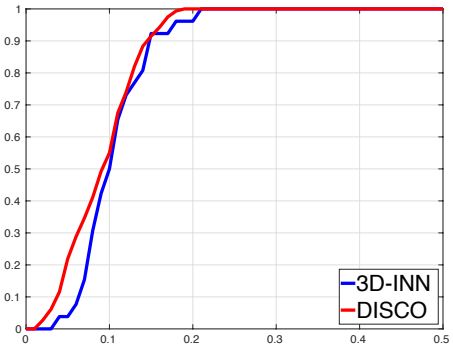
[5]Please refer to [32] for the explanation of low accuracy of "GT CAD".



(a) IKEA Sofa



(b) IKEA Chair



(c) IKEA Bed

Figure 4.14: 3D PCK (RMSE[73]) curves of DISCO and 3D-INN on sofa (Figure 4.14a), chair (Figure 4.14b) and bed (Figure 4.14c) classes of IKEA dataset. In each figure, X axis stands for α of PCK and Y axis represents the accuracy.

4.4.3.4 IKEA

In this section, we evaluate DISCO on the IKEA dataset [33] with 3D keypoint annotations provided by [73]. One question remaining for the DISCO network is whether it is capable of learning 3D object geometry for multiple object classes simultaneously. Therefore, we train a single DISCO network from scratch which jointly models three furniture classes: sofa, chair and bed. At

Method	Sofa		Chair		Bed	
	Recall	PCK	Recall	PCK	Recall	PCK
3D-INN	**88.0**	31.0	87.8	41.4	**88.6**	42.3
DISCO	84.4	**37.9**	**90.0**	**65.5**	87.1	**55.0**

Table 4.7: Average recall and PCK[$\alpha = 0.1$] accuracy(%) for 3D structure prediction on the sofa and chair classes on IKEA dataset.

test time, we compare DISCO with the state-of-the-art 3D-INN[73] on IKEA. Since 3D-INN evaluates the error of 3D structure prediction in the object canonical pose, we align the PCA bases of both the estimated 3D keypoints and their groundtruth. Table 4.7 reports the PCK[$\alpha = 0.1$] and average recall[73] (mean PCK over densely sampled α within $[0, 1]$) of 3D-INN and DISCO on all furniture classes. The corresponding PCK curves are visualized in Figure 4.14. We retrieve PCK accuracies of 3D-INN on the IKEA dataset from its publicly released results. DISCO significantly outperforms 3D-INN on PCK by $6.6\%, 24.1\%, 12.7\%$ on sofa, chair and bed respectively, which means that DISCO obtains more correct predictions of keypoint locations than 3D-INN. This substantiates that direct exploitation of the rich visual details from images adopted by DISCO is critical to infer more accurate and fine-grained 3D structure than lifting sparse 2D keypoints to 3D shapes like 3D-INN. However, DISCO is inferior to 3D-INN in terms of average recall on the sofa and bed class. As shown in Figure 4.14a, the incorrect predictions by DISCO deviate more from the groundtruth than 3D-INN. This is mainly because 3D predicted shapes from 3D-INN are constrained by shape bases so even incorrect estimates have realistic object shapes when recognition fails. Moreover, our 3D keypoint labeling for the sofa CAD models is slightly different from [73]. We annotate the corners of reachable seating areas of a sofa

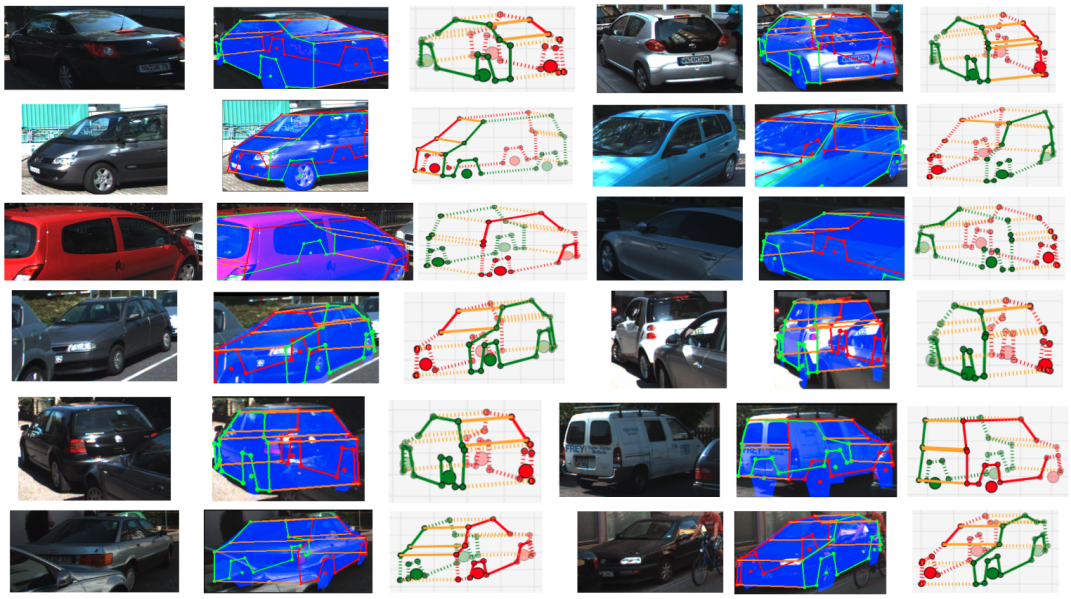


Figure 4.15: Visualization of 2D/3D prediction, visibility inference and instance segmentation on KITTI-3D. Circles and lines represent keypoints and their connections. Red and green indicate the left and right sides of a car, orange lines connect two sides. Dashed lines connect keypoints if one of them is inferred to be occluded. Light blue masks present segmentation results.

while IKEA labels the corners of the outer volume parallel to the seating area We conclude that DISCO is able to learn 3D patterns of object classes other than the car category and shows potential as a general-purpose approach to jointly model 3D geometric structure of multiple objects *in a single model*.

4.4.3.5 Qualitative Results

We visualize example predictions from DISCO on KITTI-3D (Figure 4.15) and PASCAL VOC (Figure 4.16). From left to right, each column shows the original object image, the predicted 2D object skeleton with instance segmentation and the predicted 3D object skeleton with visibility. We show the results under no occlusion, truncation, multi-car occlusion and other occluders (e.g.

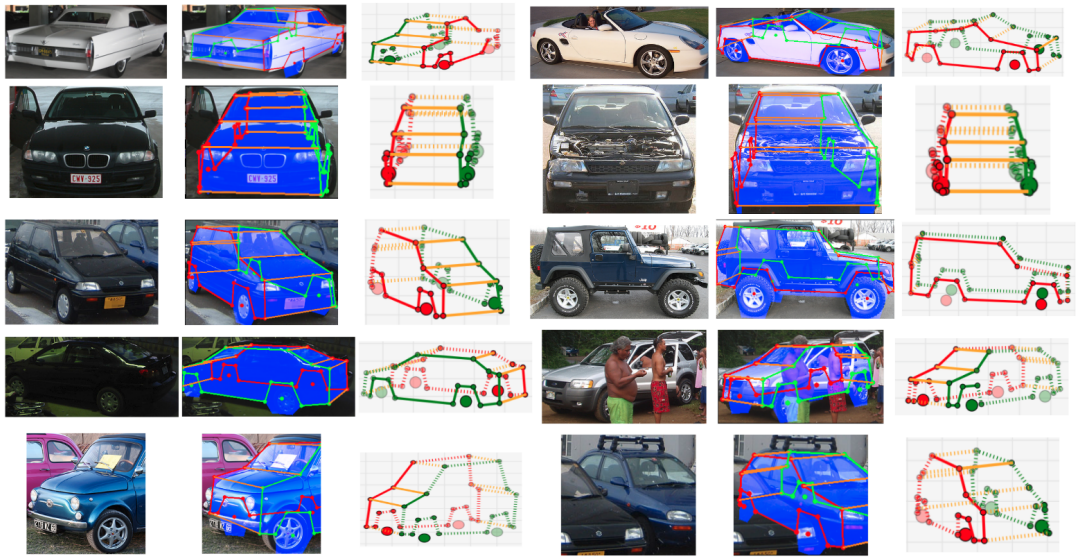


Figure 4.16: Visualization of 2D/3D prediction, visibility inference and instance segmentation on PASCAL VOC.

human). We observe that DISCO is able to localize 2D and 3D keypoints on real images with complex occlusion scenarios and diverse car models such as sedan, SUV and pickup. Moreover, the visibility inference is mostly correct. These capabilities highlight the potential of DISCO as a building block for holistic scene understanding in cluttered scenes. In failure cases shown in Figure 4.17, the left car is mostly occluded by another object and the right one is severely truncated and distorted in projection. We may improve the performance of DISCO on these challenging cases by exploiting more sophisticated data simulated with complex occlusions [180] and finetuning DISCO on real data.

In addition, we qualitatively compare 3D-INN and DISCO on three categories in IKEA dataset in Figure 4.18. For the chair, 3D-INN fails to delineate

Figure 4.17: Two failure cases of 2D/3D keypoint localization on the car category.

the inclined seatbacks in the example images while DISCO being able to capture this structural nuance. For the sofa, DISCO correctly infers the location of sofa armrest whereas 3D-INN merges armrests to the seating area or predicts an incorrect size of the seatback. Finally, DISCO yields better estimates of the scale of bed legs than 3D-INN. We attribute this relative success of DISCO to direct mapping from image evidence to 3D structure, as opposed to lifting 2D keypoint predictions to 3D.

4.5 Image Classification

In this section, we show another application of our generic deep supervision framework for image classification on CIFAR100 [181]. We exploit coarse-grained class labels (20-classes) from CIFAR100 [181] to assist the fine-grained recognition on 100 object classes. This scenario is exactly the same as the discrete intermediate concepts illustrated in Figure 4.1.

4.5.1 Network Architecture

We use a network architecture similar to the one described in Section 4.4.2 but with only 20 layers. Figure 4.19 shows the details of the convolutional network. The number of filters are 128, 256 and 512 for layers of 1-5, 6-10 and 10-20 respectively. Downsampling via striding 2 on convolutional layer is performed

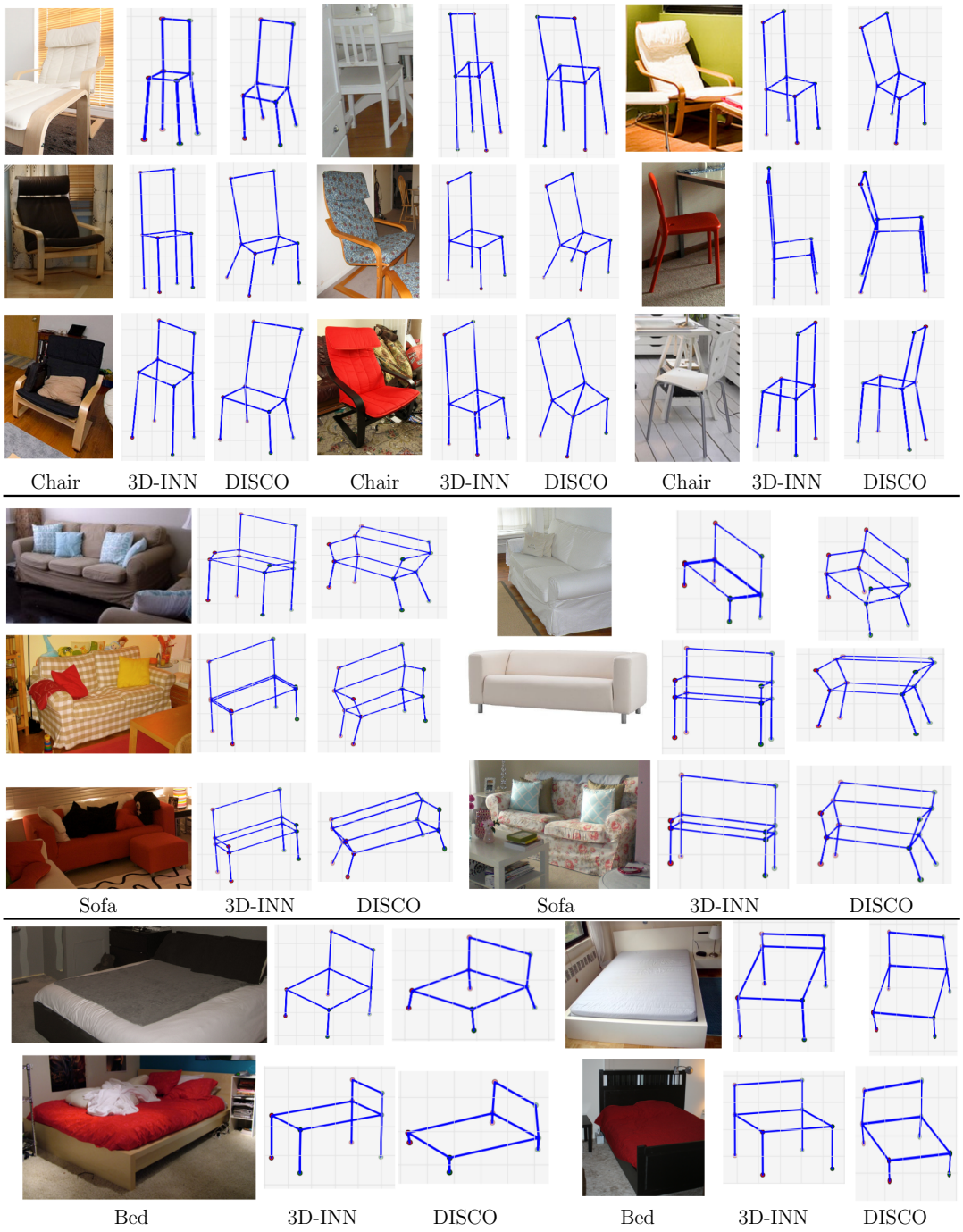


Figure 4.18: Qualitative comparison between 3D-INN and DISCO for 3D stricture prediction on IKEA dataset.

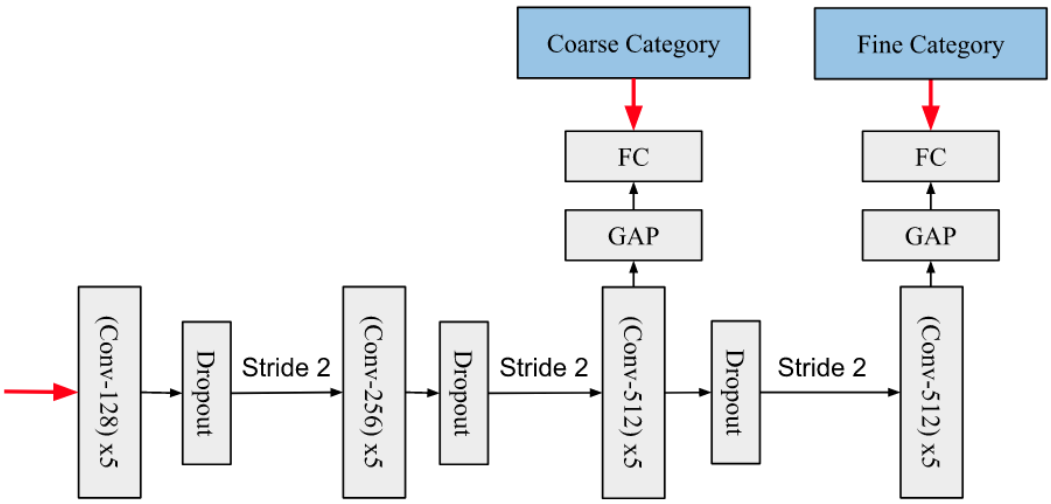


Figure 4.19: The network architecture deeply supervised by coarse-grained category labels for fine-grained classification on CIFAR100.

at layer 6, 11 and 16. The 20-class coarse-grained category label supervises at layer 15. Dropout layers are deployed on layer 5, 10 and 15. Global average pooling (GAP) layers are used to summarized the convolutional response and yield to the global descriptor for final classification on both category levels.

4.5.2 CIFAR-100

Most existing methods directly learn a model for fine-grained classification task while ignoring coarse-grained labels. In contrast, we leverage coarse-grained labels as an intermediate concept in our formulation. As such, coarse-grained class labels used as intermediate concepts are able to improve fine-grained recognition performance, which further validates our deep supervision strategy.

Table 4.8 compares the error of DISCO with state-of-the-art and variants

Methods	Error(%)
DSN[35]	34.57
FitNet, LSUV[182]	27.66
ResNet-1001[45]	27.82
pre-act ResNet-1001[183]	22.71
plain-single	23.31
plain-all	23.26
DISCO-random	27.53
DISCO	**22.46**

Table 4.8: Classification error of different methods on CIFAR100. The first four are previous methods and "pre-act ResNet-1001" is the current state-of-the-art. The remaining four are results of DISCO and its variants.

of DISCO. We use plain-single and plain-all to denote the networks with supervisions of single fine-grained label, and both labels at last layer, respectively. DISCO-random uses a (fixed) random coarse-grained class label for each training image. Note that we fix the random coarse-grained labels of all training data during training once we generate them. We observe that plain-all achieves roughly the same performance as plain-single, which replicates our earlier finding (Section 4.4.3.1) that intermediate supervision signal applied at the same layer as the main task helps relatively little in generalization. However, DISCO is able to reduce the error of plain-single by roughly 0.6% using the intermediate supervision signal. These results support our derivation of Proposition 1 and Proposition 2 in Section 4.3.3. Further, DISCO-random is significantly inferior to DISCO as a random intermediate concept makes the training more difficult. Finally, DISCO slightly outperforms the current state-of-the-art pre-act ResNet-1001[183] on image classification but with only half of the network parameters compared with [183].

4.6 Conclusion

Visual perception often involves sequential inference over a series of intermediate goals of growing complexity towards the final objective. In this chapter, we have employed a probabilistic framework to formalize the notion of intermediate concepts which points to better generalization through deep supervision, compared to the standard end-to-end training. This inspires a CNN architecture where hidden layers are supervised with an intuitive sequence of intermediate concepts, in order to incrementally regularize the learning to follow the prescribed inference sequence.

We practically leveraged this superior generalization capability to learn the object geometry and localize object shape keypoints. To cope with the scarcity of 3D annotation, we exploit synthetic training images with complex multiple object configurations for learning shape patterns. Our experiments demonstrate that out approach outperforms current state-of-the-art methods on 2D and 3D landmark prediction on public datasets, even with occlusion and truncation. We also apply deep supervision to fine-grained image classification and showed significant improvement over single-task as well as multi-task networks on CIFAR100. Finally, we have presented preliminary results on jointly learning 3D geometry of multiple object classes within a single CNN.

The present method is unable to model highly deformable objects as well as topologically inconsistent object categories such as buildings. These problems may be approachable by gaining access to more versatile datasets, and by improving output representation. Another direction of future work is to extend the current architecture for a unified framework for learning shared

representations for diverse object classes. We also see wide applicability of deep supervision, even beyond computer vision, in domains such as robotic planning, scene physics inference and generally wherever deep neural networks are being applied. One more interesting direction is to extract label relationship graphs from the CNN supervised with intermediate concepts, as opposed to explicitly constructed Hierarchy and Exclusion graphs [158].

Chapter 5

Multi-Class Multi-View Object Pose Estimation

In this chapter, we explore the problem of recovering six Degree of Freedom (6-DoF) pose of rigid objects. It is a core problem for a wide range of applications including robotic manipulation, navigation, augmented reality and autonomous driving. We first introduce a baseline single-view pose estimation system based on a geometry matching algorithm ObjRecRANSAC [4] in Section 5.1. Subsequently, this baseline system is extended to the multi-view scenario via a novel probabilistic semantics fusion framework, as detailed in Section 5.2. Finally, we present a principled end-to-end learning architecture for predicting 6-DoF object pose given a single RGB or RGB-D image. This learning-based approach is further boosted by an efficient multi-view hypothesis selection scheme which resolves single-view ambiguity.

5.1 Geometry Based Approach

In this section, we first briefly review ObjRecRANSAC which is one representative 6-DoF pose estimation algorithm based on geometry matching. This geometry based object registration technique is coupled with the semantic segmentation algorithms (described in Section 3.4 and Section 3.5) for object pose estimation. We present two baseline experiments on LN-66 in Section 5.1.3.2 and JHUScene-50 in the Section 5.1.3.3.

5.1.1 ObjRecRANSAC

ObjRecRANSAC is an efficient pose estimation algorithm originally reported in [4]. We used it as one option of object registration in our pipeline due to its efficiency and robustness to complex occlusions. The reference implementation of this algorithm is called "ObjRecRANSAC" and is available for academic use under an open-source license.[1]

ObjRecRANSAC is designed to perform fast object pose prediction using oriented point pair features $((p_i, n_i), (p_j, n_j))$ where p_i, p_j are the 3D positions of the points and n_i, n_j are their associated surface normals. In turn, a simple descriptor $f(i, j)$ is defined by:

$$f(i,j) = \begin{pmatrix} \|p_i - p_j\| \\ \angle(n_i, n_j) \\ \angle(n_i, p_j - p_i) \\ \angle(n_j, n_i - p_i) \end{pmatrix} \tag{5.1}$$

[1] See http://github.com/tum-mvp/ObjRecRANSAC.git for the reference implementation of [4].

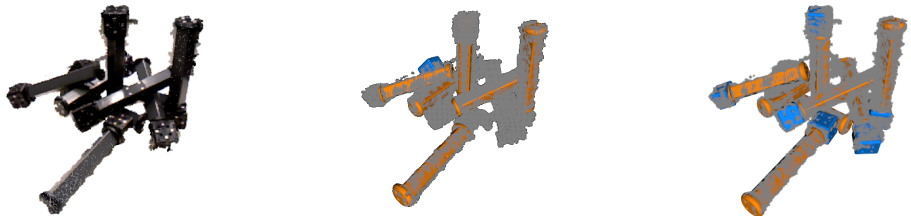


Figure 5.1: The illustration of failure cases of ObjRecRANSAC. Figures from the left to right are the testing scene, estimated poses from ObjRecRANSAC and groundtruth.

where $\angle(a, b)$ denotes the angle between a and b. Then a hash table is constructed for fast matching of point pairs from object models to the scene. We refer the reader to [4] for more details.

In ObjRecRANSAC, only oriented point pair features with fixed predefined distance d are used for RANSAC sampling. This prevents the algorithm from recognizing scenes composed of objects with significantly different characteristic lengths. If one object has a highly eccentric shape, it is best localized by sampling point pairs which span it's widest axis. This large pair separation, however, prevents any smaller objects in the scene from being recognized. Moreover, for objects situated in cluttered and occluded scenes, the probability of sampling point pairs from single object instances significantly decreases, which leads to the strong degradation in performance. One failure case of ObjRecRANSAC is shown in Figure 5.1.

From a high-level perspective, the information needed to improve the recognition accuracy in heterogeneous scenes is the object class membership. If such class membership could be determined independently from object pose, it could be used to partition the input data into independent RANSAC

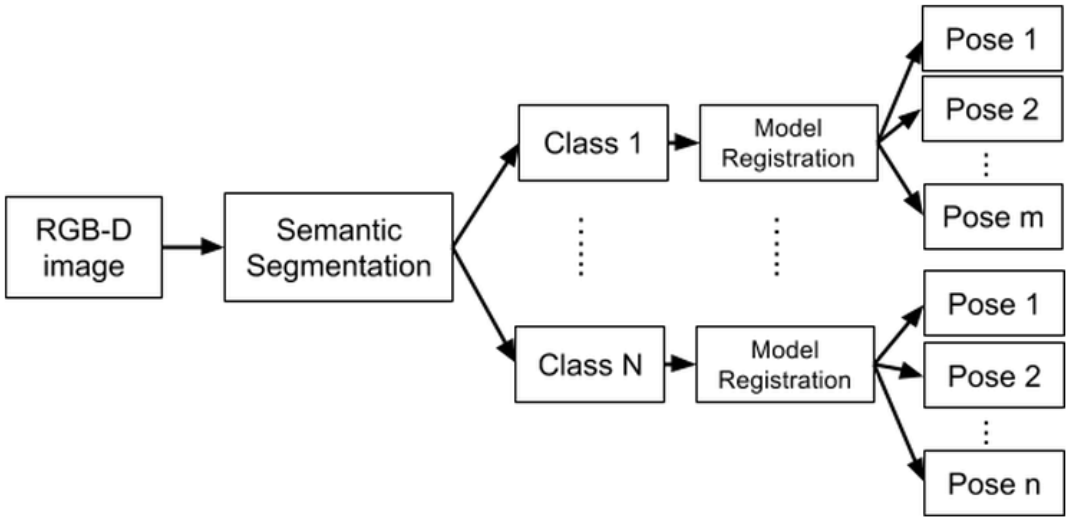


Figure 5.2: Overview of the object pose estimation framework based on semantic segmentation.

pipelines which are specifically optimally parameterized. Semantic segmentation techniques are well-suited to provide this crucial information.

5.1.2 Object Pose Estimation on Semantic Partitions

The overview of our object pose estimation framework is shown in Figure 5.2. We redesign the multi-domain pooling architecture for the semantic segmentation and subsequently apply ObjRecRANSAC registration techniques to estimate object poses for each semantic class. We initialize ObjRecRANSAC models separately for each class and the length for their oriented point pair features is set as 0.07m for all objects. Other parameters are kept the same as those reported in the original implementation [4].

To reduce the number of false positives returned from ObjRecRANSAC, a simple non-maximum suppression step is carried out to filter inaccurate pose estimation. Consider the function $Q(q, M, P)$ that indicates the confidence

score of the pose q for the model M supported by the scene cloud P:

$$Q(q, M, P) = \sum_{v_i \in M} \mathbf{1}(\min_{p_i \in P} \|T(v_i, q) - p_i\|_2 < \delta_D) \tag{5.2}$$

where v_i is the vertex in object mesh M and $T(v_i, q)$ transforms the v_i based on pose q. δ_D is a pre-set parameter. Both M and P can be represented by voxel grids, which makes function $Q(q, M, P)$ highly efficient in practice. We reject the hypothesis with lower score from any pair of hypotheses whose projected 2D intersection is more than 50% of its union. The remaining hypotheses are the final poses estimated by the algorithm for the scene.

5.1.3 Experiment

5.1.3.1 Evaluation Metric

Unlike the matching function designed for LINE-MOD[5], we introduce a more flexible matching criterion to determine whether an estimated pose is correct. In the task of object manipulation, a good pose estimation needs to achieve high matching accuracy only with respect to the 3D geometry but not the surface texture. This implies that for objects with certain symmetrical structure (rotational and/or reflective), there should exist multiple pose candidates having perfect matching to the groundtruth. Thus, we design a new distance function between two estimated poses (i.e. 3D transformation in SE(3)) T_1 and T_2 for model point cloud P_M with N 3D points uniformly

sampled from the full object mesh:

$$D(T_1, T_2; P_M) = \frac{\sum_{p_i \in P_M} \mathbf{1}(\min_{p_j \in P_M} \|T_1(p_i) - T_2(p_j)\|_2 < \delta_D)}{N} \tag{5.3}$$

where threshold δ_D controls the matching degree. Another threshold R_D is used to justify an estimated pose T with respect to the groudtruth T_g by the criterion: $D(T, T_g; P_M) \geq R_D$. We set $\delta_D = 0.01$ and $R_D = 0.7$ for all our experiments. Finally, We evaluate the performance of pose estimation algorithm by precision and recall accuracies:

$$Precision = \frac{|\{\text{true positives}\}|}{|\{\text{all predicted poses}\}|} \tag{5.4}$$

$$Recall = \frac{|\{\text{true positives}\}|}{|\{\text{all groundtruth poses}\}|} \tag{5.5}$$

5.1.3.2 LN-66

For this dataset, we use the sliding-window based semantic segmentation (Section 3.4) to partition the scene into different semantic classes. Although the semantic segmentation narrows down the space of RANSAC sampling within only a single semantic class, the ratio of inlier correspondences may be still small due to multiple adjacent or connected object instances. We first introduce two recursive pipelines that improve the performance of ObjRecRANSAC in terms of stability and the recall rate. In what follows, we denote the original ObjRecRANSAC as **B** short for Batch Matching and introduce two improved variants as **GB** and **GO**.

Greedy-Batch Matching(GB): In this approach, we run the ObjRecRANSAC recursively over the parts of the scene that have not been well explained by previous detected models. Specifically, the initial inputs to the ObjRecRANSAC are the set of segmented points P_0 that share the same class label. At the ith round of recognition ($i \geq 1$), the working space P_i is constructed by removing the points in P_{i-1} that can be explained by the detected models M_{i-1} at $(i-1)th$ round:

$$P_i = \{p \mid \min_{m \in M_{i-1}} \|p - m\|_2 > T_d \ \wedge \ p \in P_{i-1}\} \tag{5.6}$$

where T_d is the threshold (set to 0.01m) to determine the inlier. The detected models M_{i-1} are the transformed point clouds that are uniformly sampled from full object meshes. Finally, this greedy registration pipeline stops once no more instances are detected. The final set of estimated poses is the union of all previously detected poses: $M_{final} = \cup_i M_i$.

Greedy-One Matching(GO): The **GB** approach can fail to discover some object instances because false positives in early iterations can lead to false negatives later on. In order to achieve higher precision and recall detection rates, we adopt a more conservative greedy approach in which we only choose the best detected object candidate with the highest confidence score from ObjRecRANSAC as the current detected model M_i at ith round. The rest follows the same implementation as in **GB**. The simple flow charts for **B**, **GB** and **GO** are illustrated in Figure 5.3.

Next, we report the means and standard deviations(std) of precision, recall

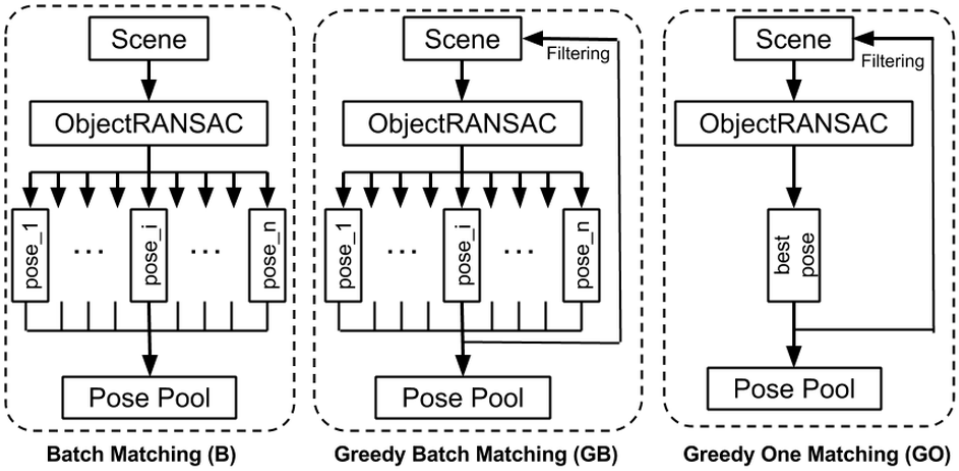


Figure 5.3: Illustration of variants of ObjRecRANSAC of **B**, **GB** and **GO**.

and F-measure [2] of our algorithm on LN-66 in Table 5.1. For comparison, we run experiments for different variants of our algorithm whose names are formatted as 'S+O'. The first entry 'S' indicates the degree of semantic segmentation used with three specific options '**NS**', '**S**' and '**GS**' as *no segmentation*, *standard segmentation* (Section 3.4) and *groudtruth*. The second entry 'O' stands for the three choices of ObjRecRANSAC including '**B**', '**GB**' and '**GO**'. Due to the randomized process in ObjRecRANSAC, we run 50 trials of each method over all testing data.

From Table 5.1, we observe that: 1) the semantic segmentation significantly improves all three RANSAC-based pose estimation methods in terms of precision/recall rates; 2) when using the segmentation computed by our algorithm, the RANSAC stage performs only 2 $\sim$ 4% behind in the final F-measure compared to using the groundtruth segmentation. 3) Both **GO** and **GB** are more accurate (higher F-measure) and stable (smaller standard deviation)

[2] F-measure is a joint measurement computed by precision and recall as $\frac{2 \cdot precision \cdot recall}{precision + recall}$

	Precision(%)	Recall(%)	F-Measure
NS+B	84.47 ± 0.36	61.75 ± 0.27	71.30 ± 0.28
NS+GB	79.88 ± 0.47	79.42 ± 0.37	79.65 ± 0.40
NS+GO	88.63 ± 0.31	83.13 ± 0.32	85.80 ± 0.30
S+B	87.77 ± 0.20	81.31 ± 0.27	84.42 ± 0.22
S+GB	91.89 ± 0.24	89.27 ± 0.19	90.56 ± 0.21
S+GO	94.50 ± 0.16	91.71 ± 0.13	93.09 ± 0.12
GS+B	97.27 ± 0.06	87.03 ± 0.14	91.87 ± 0.08
GS+GB	95.29 ± 0.10	92.33 ± 0.13	93.79 ± 0.11
GS+GO	98.79 ± 0.20	94.33 ± 0.13	96.51 ± 0.16

Table 5.1: Reported precision, recall and F-score by different methods on LN-66 dataset.

S-CSHOT	**S-Int**	**S-Det**
0.39 ± 0.12	0.13 ± 0.03	0.31 ± 0.12
B(NS)	**GB(NS)**	**GO(NS)**
0.86 ± 0.16	1.49 ± 0.40	7.69 ± 3.43
B(S)	**GB(S)**	**GO(S)**
0.85 ± 0.20	2.16 ± 0.68	4.40 ± 1.72

Table 5.2: Means and standard deviations of running times of different methods on LN-66 dataset.

than the standard ObjRecRANSAC (**B**) regardless whether they are supported by semantic labeling.

Furthermore, we show an example of comparison between different methods in Figure 5.4 and more results from **S+GB** are shown in Figure 5.5. In each sub-figure of Figure 5.4, the gray points represent the point cloud of the testing scene. The estimated poses for the "link" and "node" objects are shown in yellow and blue meshes, respectively. We can see that methods that work on semantically segmented scenes achieve noticeable improvement over the ones without scene classification. In addition, the computed semantic segmentation yields similar results ((d), (e), (f) in Figure 5.4) compared with the ground

truth ((g), (b), (i) in Figure 5.4), which shows the effectiveness of our semantic segmentation algorithm. Also, **GO** and **GB** outperform **B** whether or not semantic segmentation is used. From Figure 5.5, we can see **S+GB** could reliably detect and estimate object poses in cluttered and occluded scenes. Finer pose refinement can be made by incorporating physical constraints between adjacent objects.

Finally, Table 5.2 reports the means and standard deviations of running times of all main modules in the semantic segmentation as well as **B**, **GB**, **GO** in two contexts: (**S**) and (**NS**) indicating with and without semantic segmentation, respectively. For semantic segmentation, we evaluate all three components: CSHOT extraction (**S-CSHOT**), integral image construction (**S-Int**) and classification of sliding windows (**S-Det**). From Table 5.2, we can see that the semantic segmentation is running efficiently compared to the overall runtime of the algorithm. Furthermore, all three sub-stages can be trivially parallelized and dramatically accelerated with GPU-based implementations. We also observe that the semantic segmentation reduces the runtime of **GO(NS)** by half because it decreases the number of RANSAC hypotheses in this greedy approach. For pose estimation, two proposed greedy approaches **GB** and **GO** are slower than the standard one **B** due to multiple runs of ObjRecRANSAC. Additionally, **GB** performs only slightly worse than **GO** (shown in Table 5.1) while being much more efficient. These times were also computed for the CPU-based implementation of ObjRecRANSAC, and did not use the GPU-accelerated implementation, which is already available under the same open-source license. The choice of these three methods in

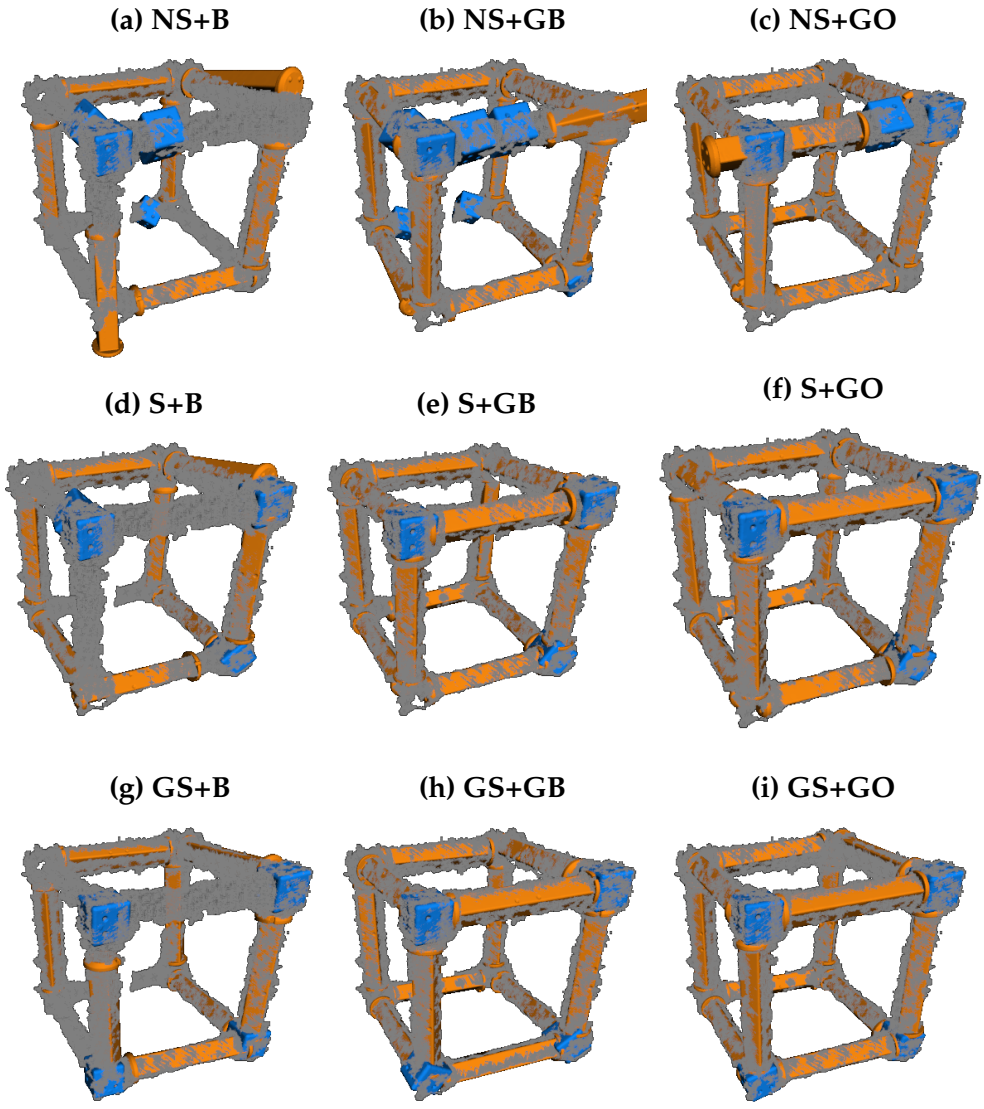


Figure 5.4: An example of the comparison of the estimated poses by different methods.

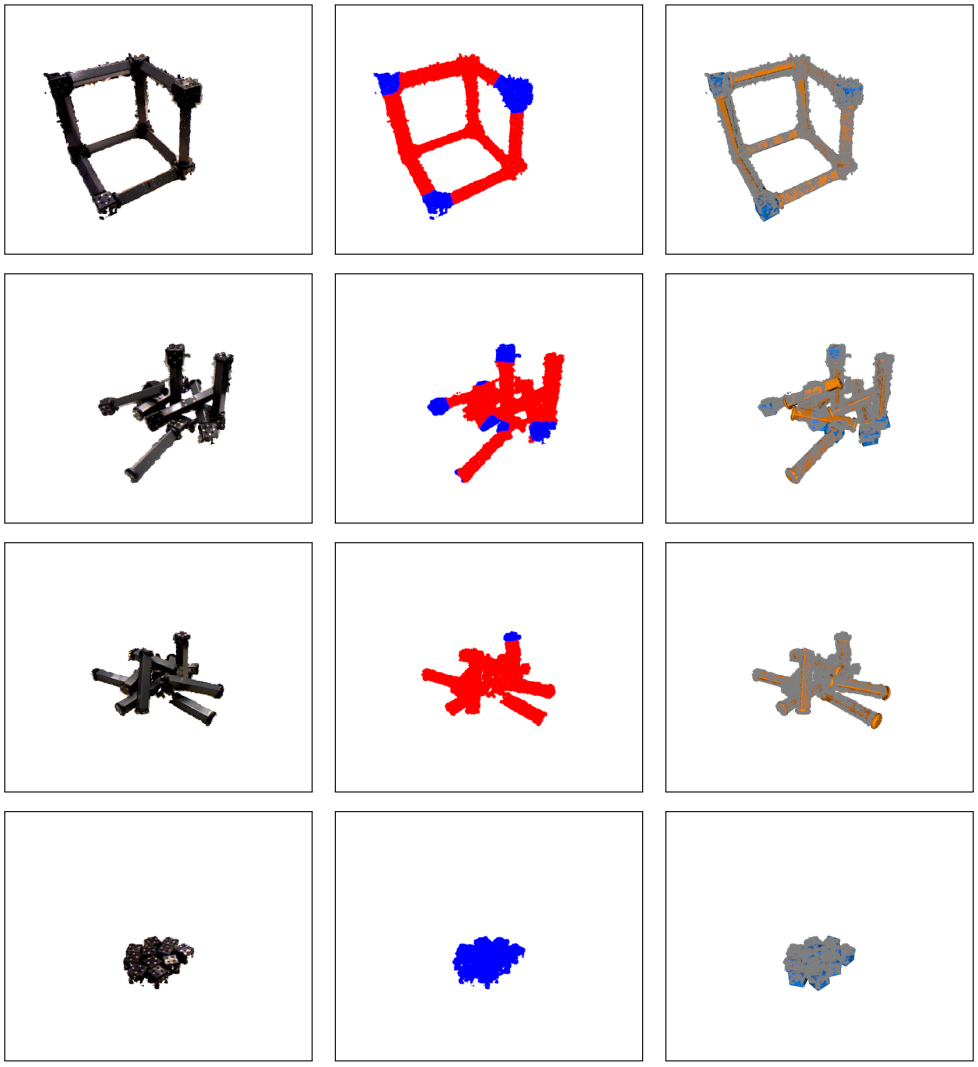


Figure 5.5: Example results of **S+GB** on LN-66. The left, middle and right columns show the testing scenes, segmentation results and estimated poses, respectively.

practice can be decided based on the specific performance requirements of a given application.

Although the overall runtime of the entire perception system takes more than 1s even without the semantic segmentation, this may not be a main issue to integrate our algorithm into a real-time robotic system. First, GPU-based parallel programming techniques could significantly speed up the current implementation. Second, standard object tracking methods [6] can be initialized by our algorithm to track object poses in real time and reinitialized when they fail.

5.1.3.3 JHUScene-50

We use the hierarchical semantic segmentation (Section 3.5) to partition the scene into different semantic classes.

We test the overall performance of our method for instance pose estimation. Following the same metric used in Section 5.1.3.2, we decide the correctness of an estimated pose if the transformed object mesh has more than 70% surface overlap with the groundtruth. This criterion does not measure the matching between surface textures because the accurate prediction of 3D object occupancy is the dominant factor in most of perception scenarios such as the object manipulation. In addition, multiple optimal solutions may be valid for objects with symmetrical structures in shape and texture under certain partial views. Given a test frame, we calculate the precision/recall of all estimated poses as the measurement of pose estimation performance.

Algorithm	Precision / Recall					
	Office	Labpod	Barrett	Box	Shelf	Overall
SHOT+Hough Voting[184]	0.02 / 0.91	0.01 / 0.41	0.04 / 0.53	0.02 / 0.66	0.07 / 1.30	0.32 / 0.76
Hypotheses Verification[38]	3.2 / 2.1	2.8 / 2.1	1.7 / 1.1	3.6 / 0.55	7.3 / 4.5	3.7 / 2.1
Vanilla ObjRecRANSAC[4]	1.4 / 4.3	1.3 / 4.7	1.8 / 7.2	0.7 / 2.6	2.4 / 10.3	1.5 / 5.8
Foreground Seg. + ObjRecRANSAC	41.8 / 45.2	38.9 / 41.0	41.8 / 43.4	43.3 / 43.4	48.3 / 48.2	42.8 / 44.3
Object Seg. + ObjRecRANSAC	84.8 / 78.3	78.9 / 69.4	78.4 / 67.0	80.4 / 70.4	85.9 / 80.3	81.7 / 72.7
Groundtruth Seg. + ObjRecRANSAC	96.3 / 95.8	89.6 / 86.2	96.5 / 95.1	94.3 / 93.2	96.3 / 95.3	94.6 / 93.1

Table 5.3: Reported precision and recall of the estimated object poses by different algorithms in 5 indoor contexts. "Seg." is short for segmentation.

We report the average precision and recall rates of different pose estimation algorithms in Table 5.3. We can see that pure local matching methods [184, 38, 4] barely work on JHUScene-50 while our method "Object Segmentation+ObjRecRANSAC" being able to retrieve 73.1% correct poses with 81.6% precision. Moreover, we analyze how semantic segmentation helps the model registration by running ObjRecRANSAC within four different scenarios: the raw scene (no semantics), extracted foreground regions, classified object regions and groundtruth segmentation. From Table 5.3, the vanilla ObjRecRANSAC [4] (no semantics) performs much worse than the other three variants of our method with semantics support at different degrees. The foreground-based ObjRecRANSAC significantly degrades the performance of the object-based ObjRecRANSAC by roughly 40% on precision and 30%

on recall. Last, the groundtruth segmentation enables the ObjRecRANSAC to achieve the highest precision and recall that are both above 93% on average. In conclusion, the more input of the correct semantics information from the scene, the better performance of pose estimation. This supports the basic motivation for our semantics-based pose estimation framework. The high precision/recall performance (94.6%/93.1%) based on the groundtruth segmentation shows a promising future of improving the current semantic parsing method to approach the perception requirements of real robotic systems.

Error Analysis and System Runtime. We first report that our algorithm could yield at least one correct object pose in 99.5% of all 5000 frames. That means only one in 200 times that a robot would not be able to interact with any object in a scene. Figure 5.6 shows some qualitative results of our semantic segmentation and pose estimation algorithms. More results can be found in the supplementary video. We can see from subfigures (a) to (g) in Figure 5.6 that ObjRecRANSAC is able to yield correct object poses given an imperfect semantic segmentation. One failure case is also demonstrated in the bottom-right corner of Figure 5.6. The silver hammer and yellow mallet behind it are not detected due to their similar appearances to other objects as well as the background clutter. One way to improve the current semantic classifiers is to retrain SVMs with the false positives and negatives in the training set returned from the original classifiers.

The current runtime of the whole system is around 5 seconds per scene on average. All tests are performed on a desktop with Intel Xeon CPU E5-2690.

The most time consuming parts are the CSHOT and FPFH feature computation which can be dramatically accelerated with GPU-based implementations. Additionally, the estimated poses from our algorithm could be used to initialize a locally-stable object tracking system to continuously estimate the object poses in a dynamic scene.

5.2 Multi-View Pose Estimation via Geometry Matching on Dense SLAM

Major challenges in single-view perception are partial or complete occlusion among object instances, large viewpoint variations of the same object class, and similar appearances shared across different semantic categories. These often occur in densely cluttered scenes, where multiple objects are in close contact and placed over a complex background.

As motivated in Section 1.1.3, one promising solution to overcome the aforementioned problems is to fuse semantic predictions from different viewpoints, taking advantage of the fact that multiple scene observations are often available in real robotic perception scenarios. Recently, various dense SLAM systems such as KinectFusion [21] have emerged for real-time dense 3D reconstruction from consecutive views. They offer fast and reliable camera pose estimation to associate perception results from different frames and establish a geometrically consistent 3D scene model. Such a scene model can represent the foundation for robustly handling occlusions and for carrying out object detection by aggregating object evidence from different viewpoints.

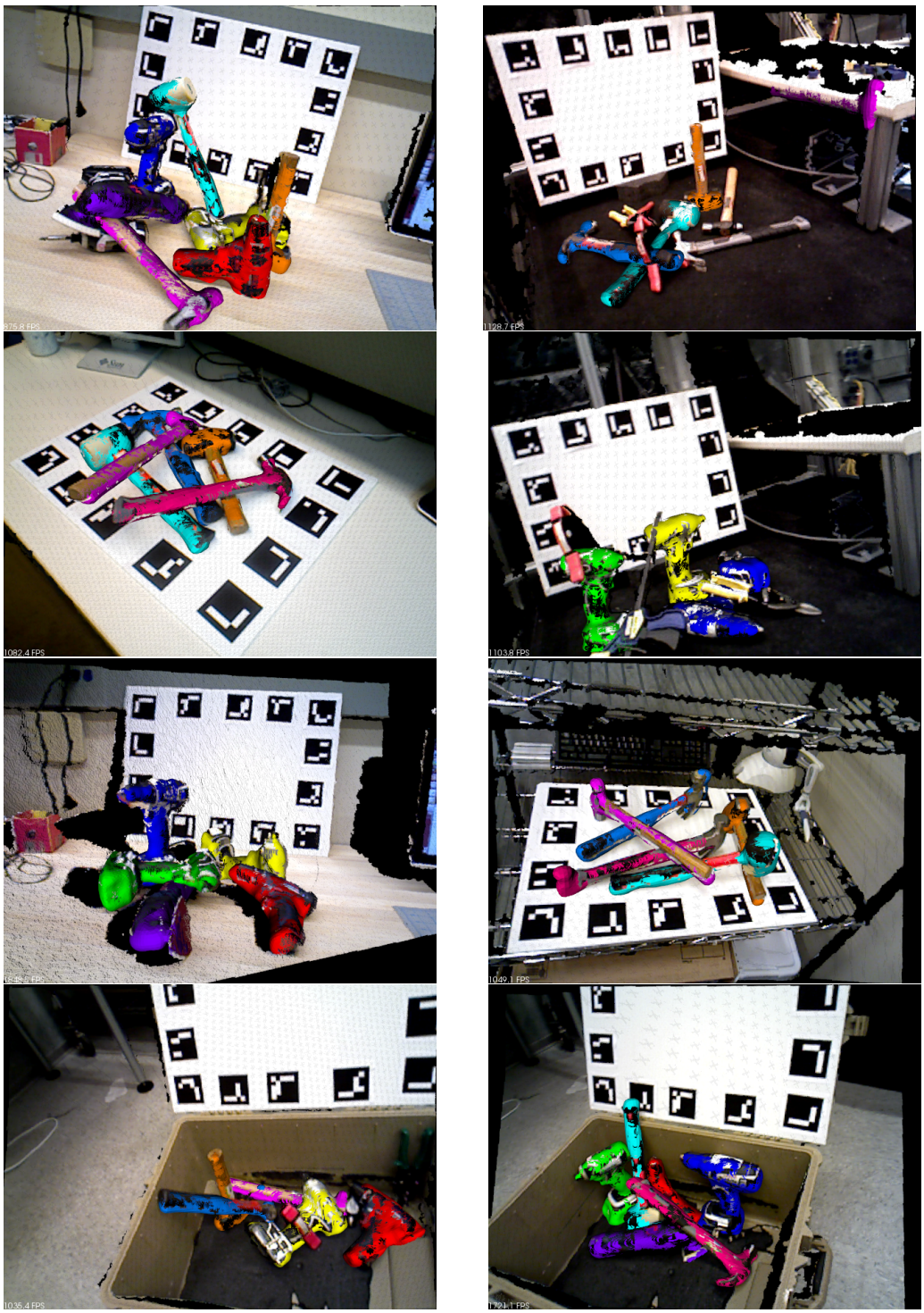


Figure 5.6: Visualization of the object pose estimation result on JHUSecne-50. Each estimated pose is highlighted by a unique color.

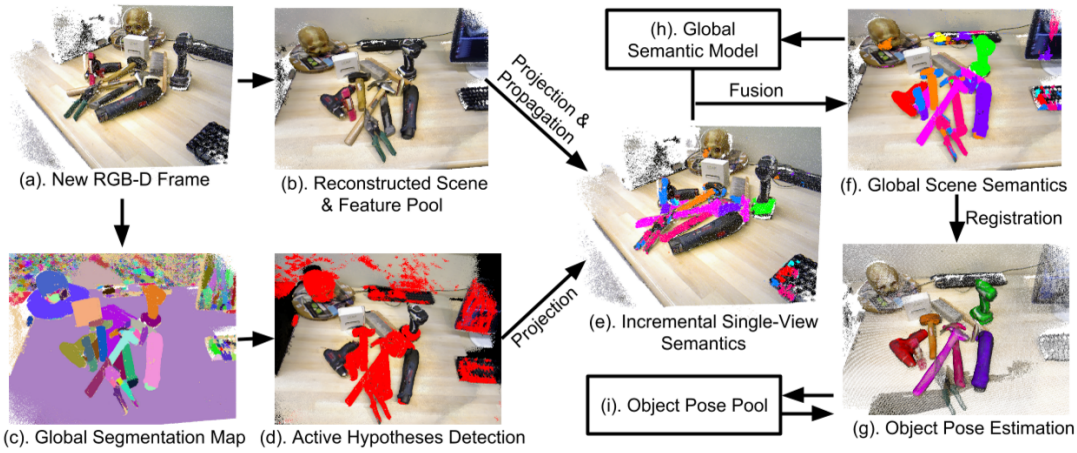


Figure 5.7: Overview of the incremental scene understanding framework. The different colors in (c) indicate hypotheses for objects and scene structures on the reconstructed scene. The red regions in (d) show the active hypotheses. Each colorized region in (e), (f) and (g) represents the segmentation of one specific semantic class or object pose in consistent colorization.

In this section, we formulate a generic SLAM-enhanced scene understanding framework that incrementally exploits scene cues including scene semantic labels, instance locations and 6-DoF object poses. To do so, we first present a probabilistic semantic inference algorithm which predicts semantic labels for the temporally evolving hypotheses returned from our incremental segmentation system [22]. Each hypothesis is tracked over time and the Maximum A Posterior (MAP) estimation is performed over the ensemble of all hypotheses resulting from this and previous time steps. Second, we show that state-of-the-art RGB-D features (shown in Chapter 3) can be efficiently computed by propagating local filter responses via the online reconstructed scene model given the current camera pose. Third, We quantitatively and qualitatively evaluate our method on JHUSEQ-25 and demonstrate significant improvement over the single-frame based methods in terms of both accuracy and speed.

5.2.1 Overview

In this section, we first give a brief overview of the incremental scene understanding framework. Figure 5.7 illustrates the pipeline of the semantic segmentation and object pose estimation, as well as the interaction between major components in this framework. Given a new frame captured by a moving RGB-D sensor (Figure 5.7.(a)), the reconstructed scene point cloud (Figure 5.7.(b)) and the corresponding proposals for objects and scene structures (i.e. Global Segmentation Map in Figure 5.7.(c) and in Section 5.2.2) are updated via the incremental hypothesis generation system introduced by [22]. Next, active scene segments (Figure 5.7.(d)) with 3D points which are newly inserted or removed from the current frame are detected. Then, they are further projected back to the current frame based on the estimated camera pose from SLAM. In turn, we apply a probabilistic fusion scheme (in Section 5.2.3 to predict semantic labels for active hypotheses (Figure 5.7.(e)), and then update the global semantic model by integrating these predictions into the previous model (Figure 5.7.(f)). Finally, we run ObjRecRANSAC [4] to register object models into the scene regions with changed semantic labels and update the object pose pool with new estimated poses (described in Section 5.2.3.4).

5.2.2 Review of Incremental Hypothesis Generation

In the following, we review one previous work on the incremental hypothesis generation [22]. Hypothesis generation is carried out by performing unsupervised incremental segmentation on top of a dense SLAM algorithm. Each segment incrementally computed by this approach is considered as

a possible hypothesis of an object or scene structure, which will be successively assigned to a semantic category and integrated in our framework. At each time step, the output of incremental hypothesis generation is a Global Segmentation Map (GSM) that includes a set of segments or hypotheses: $H^t = \{h_1^t, \cdots, h_n^t\}$. Each hypothesis is defined as $h_i^t = \{p_{i1}^t, ..., p_{in_i}^t\} \in H^t$, where $p_{ij}^t = \{x_{ij}^t, y_{ij}^t, z_{ij}^t, r_{ij}^t, g_{ij}^t, b_{ij}^t\}$ $(1 \leq j \leq n_i)$ is the j-th 3D point with associated RGB value in segment h_i^t. H^t is updated at every new input frame, by adding new segments and/or merging old ones, as explained in the following.

The first stage is the SLAM reconstruction using [185]. This approach first estimates the camera pose of a moving sensor, then it reconstructs the scene as a point-based representation with normals, each point being augmented with information corresponding to the radius and confidence of the local surface. Each depth map at current time step is incrementally merged into the global model by means of the estimated camera pose, by updating the section of the global volume that is visible from the current viewpoint.

Next, the following four steps are carried out to maintain hypotheses in GSM that will be in turn processed by the incremental semantic segmentation stage (in Section 5.2.3). First, each depth map is segmented by extracting smooth 3D regions in which neighboring depth points contains with small normal angles. Successively, to enforce label coherency between the segments of the current frame and those in the GSM, the currently visible segments in the GSM are propagated to the current depth map by means of the estimated camera pose obtained via SLAM, and compared with those on the current frame based on their 3D overlap. When two segments show a remarkable

overlap (regulated by a threshold), the GSM segment transfers its label to the current frame, this yielding a coherent label map denoted as Propagated Label Map (PLM). Third, pairs of segments on the PLM that correspond to the same 3D surface are detected and merged, still by means of their geometric overlap. Finally, the labels of the GSM are updated with the labels computed from the PLM.

For each $h_i^t \in H^t$, the corresponding merging set is denoted as $\mathcal{M}_i^t$ which contains the set of hypotheses at $t-1$ that obtain label i on the GSM at time t. In other words, each $h_j^{t-1} \in \mathcal{M}_i^t$ is merged into h_i^t at time t. Since the frame-wise depth map segmentation can be noisy, a confidence scheme is used so that the update process is carried out only when a certain segment is consistent over a few consecutive frames, thus avoiding wrong label propagation to the GSM.

5.2.3 Incremental Semantic Segmentation

The incremental semantic segmentation temporally learns the scene semantics based on the evolving object and scene proposals H^t generated by the hypothesis generation algorithm in 5.2.2. We track the dynamic process related to the merging of different hypotheses by means of a tree-based structure referred to as Temporal Hypothesis Tree (THT). The THT holds the current hypothesis and its compositional parts that have been merged into it in previous time steps. Thus, the THT represents the merging history of each hypothesis.

To predict semantic label of $h_i^t \in H^t$, we carry out semantic classification by representing each scene segment h_i^t by the state-of-the-art RGB-D feature

for object instance classification [29] . A linear Support Vector Machine (SVM) is trained to compute the probabilities of the pre-defined semantic classes for each node in the THT. Finally, a probabilistic inference scheme is applied to predict the semantic label for h_i^t by incorporating the semantic responses at different depths of the corresponding THT.

5.2.3.1 Temporal Hypothesis Tree

Consider a THT $\mathcal{T}_i^t = \{I_j^{t_j}\}$ associated with the ith hypothesis h_i^t on GSM, where $I_j^{t_j}$ is an internal node in $\mathcal{T}_i^t$. Each internal node $I_j^{t_j} = \langle h_j^{t_j}, C_j^{t_j} \rangle$ is composed of a hypothesis $h_j^{t_j}$ formed at time step $t_j \leq t$ and its children $C_j^{t_j}$ which contains the hypotheses that belong to the corresponding merging set $\mathcal{M}_j^{t_j}$ for $h_j^{t_j}$:

$$C_j^{t_j} = \{I^{t_j-1} = \langle h^{t_j-1}, C^{t_j-1} \rangle \mid h^t \in \mathcal{M}_j^{t_j}\} \tag{5.7}$$

Additionally, the root node of $\mathcal{T}_i^t$ is I_i^t which contains the current hypothesis h_i^t and its children. When $t = 0$, we initialize the $\mathcal{T}_i^0$ by the segmented region h_i^0 on GSM computed at the first frame: $\mathcal{T}_i^0 = \{\langle h_i^0, \emptyset \rangle\}$. In turn, given a new RGB-D frame at time t, each $\mathcal{T}_i^t$ for h_i^t on current GSM [3] is incrementally developed from the ensemble of trees $\{\mathcal{T}_i^{t-1}\}$ constructed at time $t-1$. By doing so, we first acquire the merging set $\mathcal{M}_i^t$ (defined in Section 5.2.2) for h_i^t. Then, we construct $\mathcal{T}_i^t$ by connecting the new root node I_i^t to the relevant set of THTs $\widetilde{\mathcal{T}_i^t} = \cup_k \mathcal{T}_k^{t-1}$ where the hypothesis h_k^{t-1} in the root node I_k^{t-1} of each

[3] This means h_i^{t-1} is not merged into any other segment at time t

$\mathcal{T}_k^{t-1}$ belongs to $\mathcal{M}_i^t$. The root node I_i^t is constructed as follows:

$$I_i^t = \langle\ h_i^t, C_j^t = \{ I_k^{t-1} = \langle h_k^{t-1}, C_k^{t-1} \rangle \}\ \rangle\ \text{ s.t. } h_k^{t-1} \in \mathcal{M}_i^t \tag{5.8}$$

Thus, the current THT for h_i^t is formed:

$$\mathcal{T}_i^t = \{ I_i^t \} \cup \widetilde{\mathcal{T}_i^t} \tag{5.9}$$

For efficiency, if the size of h_i^t does not change too much due to the small viewpoint difference between consecutive frames, the corresponding THT is simply inherited from the previous one: $\mathcal{T}_i^t = \mathcal{T}_i^{t-1}$. In our implementation, we use a threshold γ to set active flags for hypotheses which satisfy $\frac{||h_i^t| - |h_i^{t-1}||}{|h_i^t|} \leq \gamma$ and subsequently extract features for these active segments for the semantic classification (shown in Figure 5.7.(d)). If the ith hypothesis h_i^{t-1} is merged into another hypothesis at time t (e.g. the ith hypothesis is not proposed from GSM at time t), we stop to generate its THT $\mathcal{T}_i^t$. We summarize the THT learning algorithm in Algorithm 2.

5.2.3.2 Probabilistic Inference

Once we obtain THT $\mathcal{T}_i^t$ at time t, we employ the incremental Maximum A Posteriori (MAP) estimation to predict the semantic class label $\widehat{y_i^t}$ of h_i^t. Given a pre-defined semantic class set $\mathcal{S}$, the objective of the incremental MAP

Algorithm 2 Incremental Learning of Temporal Hypothesis Tree

Input previous THTs $\{\mathcal{T}_i^{t-1}\}$, current hypotheses $\{h_i^t\}$ from GSM and parameter γ.
for each current hypothesis h_i^t **do**
  if $\frac{||h_i^t|-|h_i^{t-1}||}{|h_i^t|} > \gamma$ **then**
    Obtain hypotheses $\mathcal{M}_i^t$ that are merged into h_i^t
    Construct the current root node I_i^t by Equation 5.8.
    Construct $\mathcal{T}_i^t$ by Equation 5.9.
  else
    $\mathcal{T}_i^t = \mathcal{T}_i^{t-1}$
  end if
end for
Output updated THTs $\{\mathcal{T}_i^t\}$

semantic prediction is formulated as follows:

$$\begin{aligned}
\widehat{y_i^t} &= \underset{y \in \mathcal{S}}{\operatorname{argmax}}\ \mathrm{P}(y \mid \mathcal{T}_i^t = \{I_j^{t_j}\}) \\
&= \underset{y \in \mathcal{S}}{\operatorname{argmax}} \prod_{\langle j, t_j \rangle} \mathrm{P}(I_j^{t_j} \mid y, C_j^{t_j}) \\
&= \underset{y \in \mathcal{S}}{\operatorname{argmax}} \sum_{\langle j, t_j \rangle} \log \mathrm{P}(h_j^{t_j} \mid y, \mathcal{M}_i^t) \\
&= \underset{y \in \mathcal{S}}{\operatorname{argmax}}\ \log \mathrm{P}(h_i^t \mid y, \mathcal{M}_i^t) + \sum_{\langle j, t_j < t \rangle} \log \mathrm{P}(h_j^{t_j} \mid y, \mathcal{M}_j^{t_j})
\end{aligned} \tag{5.10}$$

The second step in Equation 5.10 is derived by applying the Bayes' Rule and Probability Chain Rule with the assumptions that $\mathrm{P}(y)$ and $\mathrm{P}(\mathcal{T}_i^t)$ are uniformly distributed and $I_j^{t_j}$ is conditionally independent from all other internal nodes given its children $C_j^{t_j}$. In the third step, we apply the log likelihood and replace the notation I_i^t with h_i^t because $\mathrm{P}(I_i^t) = \mathrm{P}(h_i^t)$ where $\mathrm{P}(h_i^t)$ indicates

the distribution of h_i^t in the space of multi-domain pooled features described in Section 5.2.3.3. The last step decouples the current hypothesis with its children and descendants, which demonstrates the incremental nature of this inference framework. Therefore, the data likelihood of the current hypothesis $\log \mathrm{P}(h_i^t \mid y, \mathcal{M}_i^t)$ is simply added to the sum of all previous likelihoods for the joint semantic prediction.

The $\mathrm{P}(h_i^t \mid y, \mathcal{M}_i^t)$ is computed as:

$$\mathrm{P}(h_i^t \mid y, \mathcal{M}_i^t) = \begin{cases} \mathrm{P}(h_i^t \mid y) & : \frac{||h_i^t| - |h_i^{t-1}||}{|h_i^t|} > \gamma \text{ or } t = 0 \\ \mathrm{P}(h_i^{t-1} \mid y, \mathcal{M}_i^{t-1}) & : \text{otherwise} \end{cases} \tag{5.11}$$

The first inequality in the first condition of Equation 5.11 is the same as the one we use in Algorithm 2 to determine whether to build a new THT for the current hypothesis h_i^t. The idea is that if h_i^t on GSM significantly changes from h_i^{t-1}, we reconstruct its features to compute the current likelihood term. Otherwise, it simply inherits from the likelihood of h_i^{t-1}. We note that this likelihood computation procedure is consistent with the THT construction pipeline.

5.2.3.3 Efficient Computation of Multi-Domain Pooled Features

The data likelihood $\mathrm{P}(h_i^t \mid y)$ in Equation 5.11 measures the similarity between the hypothesis h_i^t and the template for the semantic class y. We extract the multi-domain pooled features [29] to map each h_i^t to a feature space that is less sensitive to 3D rotation and preserves fine-grained visual cues [28]. The

feature construction can be decomposed into two stages: the convolution of local responses and the multi-domain pooling. In this work, we speed up the feature convolution stage by avoiding the recomputation for the unchanged parts on the globally reconstructed scene by SLAM.

We first consider the visible parts of the ith hypothesis h_i^t computed by the current camera pose as $\widetilde{h_i^t}$ so that $\widetilde{h_i^t} \subseteq h_i^t$. Three following steps are conducted to construct the feature for $\widetilde{h_i^t}$. First, we extract CSHOT [59] as the local descriptor for each 3D point $p \in \widetilde{h_i^t}$. To improve the efficiency of this step, we only recompute CSHOT for the active 3D points A_i^t that are newly inserted into h_i^t or removed from h_i^{t-1}. The final set of points $\widehat{A_i^t}$ that need to be updated is:

$$\widehat{A_i^t} = \{p \mid (\min \|p - \hat{p}\| \leq r_N) \wedge (\hat{p} \in A_i^t)\} \tag{5.12}$$

where r_N is the neighborhood radius parameter. Next, the depth and color components are decoupled and transformed to CSHOT local responses by conducting the soft encoding [144] over separately learned CSHOT filter sets. Finally, the generalized pooling scheme is applied to group the responses into pre-defined pooling regions in some pooling domains such as color. We combine the pooled features in all pooling regions and domains to form the final representation for $\widetilde{h_i^t}$.

Note that we use the visible parts $\widetilde{h_i^t}$ instead of h_i^t because the training data in our current implementation is captured from a single viewpoint at each time. We train the One-versus-All(OvA) SVM classifier for each semantic class. In our implementation, we minimize the sum of negative log likelihood in

Equation 5.10, where $-\log \mathrm{P}(\widetilde{h}_i^t \mid y)$ is equal to the SVM output score for class y. Finally, we assign $\log \mathrm{P}(\widetilde{h}_i^t \mid y)$ to $\log \mathrm{P}(h_i^t \mid y)$.

5.2.3.4 Incremental Object Pose Estimation

After the semantic segmentation, the reconstructed scene is partitioned into regions with homogeneous semantic labels. Then, we deploy the ObjRecRANSAC algorithm [4] to build the object model individually and estimate 6-DoF poses within the region that is classified into the corresponding object instance class. Different from [29], we only compute the object poses on the regions with the semantic labels that are changed from time $t-1$ to t [4]. Consider the object pool $\mathcal{O}^t = \{o_1^t, ..., o_N^t\}$ that contains N object meshes transformed by the estimated poses at time t. Next, we combine $\mathcal{O}^t$ with the previous object pose pool $\mathcal{O}^{t-1}$. Subsequently, a simple filtering scheme is applied to remove the false positives and duplicates in the joint set $\mathcal{O}^t \cup \mathcal{O}^{t-1}$. To achieve this goal, we first acquire the set of 3D points U_i^t on the reconstructed scene that can be explained by the transformed mesh o_i^t at time t.

$$U_i^t = \{p \mid \min_{v \in o_i^t} \|v - p\|_2 < \delta_D\} \tag{5.13}$$

where v is the vertex on the transformed mesh o_i^t and δ_D is the distance threshold to determine the correspondence [5]. For each $o_j^{t-1} \in \mathcal{O}^{t-1}$, we check whether it intersects with any $o_i^t \in \mathcal{O}^t$ by computing their overlapping ratio $\sigma = \frac{|U_j^{t-1} \cup U_i^t|}{\min\{|U_j^{t-1}|, |U_i^t|\}}$. If $\sigma > 0.5$, we remove the one with smaller set U. In practice, if the centroids of o_j^{t-1} and o_i^t are far away from each other, we can

[4] When $t = 0$, we consider all labeled object regions

[5] $\delta_D = 0.01$ in the current implementation.

simply skip the previous filtering process.

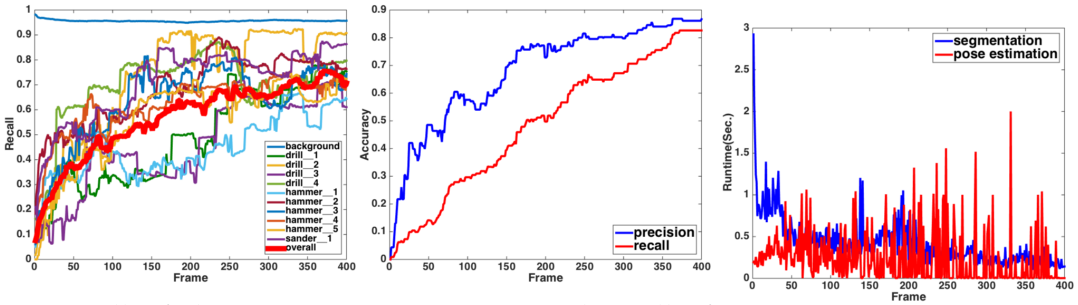
5.2.4 Experiment

In this section, we first detail the implementation of our method. Next, we provide quantitative and qualitative experimental results for semantic segmentation and object pose estimation on JHUSEQ-25 [36]. Moreover, we provide the runtime analysis of each module in our framework. The experiments demonstrate that our approach significantly improves the single-view perception performance given a stream of RGB-D images.

5.2.4.1 Implementation Details

For efficiency, we downsample each projected hypothesis $\widetilde{h_i^t}$ via octree binning with leaf size 0.003m. CSHOT descriptor is computed for downsampled points with radius 0.02m on the surface of visible parts $\widetilde{h_i^t}$ on hypothesis h_i^t. We train 100 CSHOT filters (or codewords) on UW-RGBD dataset [26] for both depth and color components separately following the same procedures described in [29]. Note that these CSHOT filters are trained on UW-RGBD dataset [26] but not JHUSEQ-25 in order to show the generalization of the proposed feature. Different from [29], we only adopt the color (in LAB) domain to pool CSHOT local responses because it is fast and sufficient to yield robust classification results within the incremental semantic segmentation framework. Level-1 to level-7 are deployed to construct pooling regions in LAB where Level-i indicates the gridding $i \times i \times i$ over three channels in LAB.

We set the depth edge threshold as 0.97 and the main level as 0 for the



(a) Recall of the semantic segmentation

(b) Precision and recall of the pose estimation

(c) Average Runtime

Figure 5.8: We show the average accuracies and runtime of our method at each frame over 25 scenes in JHUSEQ-25. Figure 5.8a shows the average recall rate of the semantic segmentation of each individual object, all objects and background. Figure 5.8b demonstrates the average precision and recall accuracies of the pose estimation. Figure 5.8c presents the average runtime of both semantic segmentation and pose estimation modules.

incremental hypothesis generation [22]. In the semantic segmentation, we follow the same two-stage process as [29] to first extract the foreground and then do the fine-grained object classification within the foreground regions. As for this purpose, we construct and update two independent THTs associated with one hypothesis for foreground/background and multi-class classification respectively. Inference results from multi-class THTs are only used when the corresponding hypothesis is classified into the foreground class.

5.2.4.2 Semantic Segmentation

To train the SVMs for foreground extraction and multi-class classification, we use the segmentation method [149] to extract parts from both object and background partial views as the training data. By doing so, our SVM models are able to classify small object segments that appear under occlusion. We use the precision and recall accuracies to measure the performance of semantic

Algorithm	Hierarchical Parsing [29]	GSM + noTHT	GSM + THT	GSM + THT + Final
drill_1	70.0 / 61.8	40.0 / 48.3	41.8 / 48.7	72.6 / 75.4
drill_2	69.7 / 80.1	52.3 / 70.0	55.8 / 73.3	61.8 / 90.7
drill_3	59.1 / 53.9	21.3 / 19.5	38.7 / 45.9	72.0 / 61.1
drill_4	89.9 / 72.0	74.7 / 63.9	92.6 / 69.8	96.8 / 80.0
hammer_1	60.7 / 40.4	53.4 / 41.1	56.2 / 44.6	71.1 / 64.4
hammer_2	61.2 / 64.5	40.6 / 58.9	41.6 / 68.3	52.6 / 76.3
hammer_3	58.8 / 62.1	37.1 / 60.7	39.9 / 63.1	54.1 / 74.4
hammer_4	39.8 / 67.8	45.5 / 55.6	52.1 / 62.4	57.3 / 71.6
hammer_5	65.5 / 59.9	35.2 / 43.4	41.9 / 56.7	69.8 / 72.4
sander_1	42.3 / 70.2	42.0 / 50.1	53.5/ 70.8	62.0 / 86.3
BG	98.8 / 97.9	99.2 / 96.2	99.3 / 95.7	99.3 / 95.7
All	57.5 / 59.3	38.2 / 45.4	45.1 / 56.7	63.8 / 70.7

Table 5.4: Reported precision and average recall rates (precision/recall) of the semantic segmentation on single-view for background (short for BG) and all objects over all scenes in JHUSEQ-25. Accuracies of variants of our method and comparative single-view methods are shown.

segmentation algorithms on a single frame[6]. The groundtruth is obtained simply by projecting the labeled object pose on 2D views.

The recall rate is more important than precision in our experiment because ObjRecRANSAC needs sufficiently large object areas for successful model registration. Therefore, we first inspect the change of recall over time. Figure 5.8a shows the average recall accuracy of each individual object, all objects and background at each frame over all 25 scenes in JHUSEQ-25. The red bold curve presents the mean accuracy across all objects at each frame. We can see that the general trend of the recall is increasing over time, although the performance fluctuates along the way due to insufficient observations to the scene. The final recalls in the last 50 frames can be above 70%, which significantly exceed

[6] precision = $\frac{|\{\text{segments}\}\cap\{\text{groundtruth}\}|}{|\{\text{segments}\}|}$, recall = $\frac{|\{\text{segments}\}\cap\{\text{groundtruth}\}|}{|\{\text{groundtruth}\}|}$

the recalls at the beginning of a sequence (less 10%). We conclude that our algorithm is able to discover more objects when more observations of the scene are available.

Table 5.4 reports the average precision and recall rates of all objects and background over all frames. "GSM+THT" is the abbreviation of our proposed method. "GSM+THT+Final" indicates only the accuracy for the final frame in a video sequence. As the table shows, "GSM+THT+Final" is much higher than "GSM+THT" by roughly 15% in both precision and recall. This again substantiates our method is able to accumulate object evidence and yield much better performance over time. "GSM+noTHT" is the variant of our method where THT is not applied and we directly classify each hypothesis based on its current SVM score. We can see that "GSM+noTHT" is inferior to "GSM+THT" by $7 \sim 10\%$ in both precision and recall, which validates the effectiveness of THT. Furthermore, the average performance of "GSM+THT" is worse than the single-view perception algorithm [29] because [29] adopts the more robust but computationally inefficient feature. However, "GSM+THT+Final" still exceeds [29] by around 10% and our method is much faster than [29] (shown in Section 5.2.4.4) in the long term.

5.2.4.3 Object Pose Estimation

Similar to [29], we define a correct estimated pose as the one which has more than 70% surface overlap with the grouthtruth pose in the same class. This criterion ignores the texture matching because 3D geometry is the dominant factor in most of perception scenarios such as the object manipulation. To

Algorithm	Precision	Recall
Hierarchical Parsing [29]	76.6	70.1
GSM + noTHT	57.8	47.0
GSM + THT	68.0	67.9
GSM + THT + Final	86.7	82.6

Table 5.5: Reported average precision and recall of the estimated poses across all objects and scenes by different algorithms on JHUSEQ-25.

evaluate the pose estimation performance, we use the precision and recall rates as the measurement.

Figure 5.8b demonstrates the plots of the average precision and recall over all objects and scenes at each frame. The rising curves in Figure 5.8b indicate that the model registration method (e.g. ObjRecRANSAC) greatly benefits from the increasing accuracy on semantic segmentation and yield above 80% accuracy in both precision and recall for the object pose estimation in densely cluttered scenes. This further validates our incremental algorithm for scene understanding.

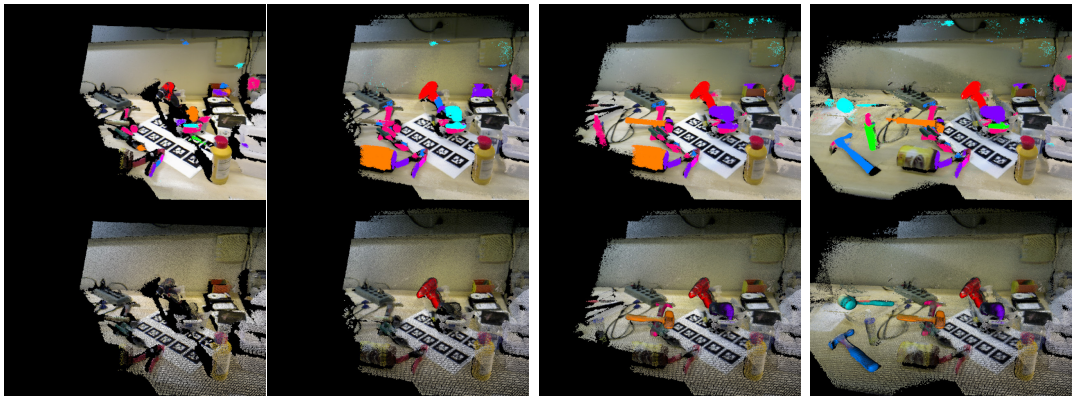
Furthermore, we report the average precision and recall of our object pose estimation algorithm over all frames, objects and scenes in Table 5.5. Similar to the semantic segmentation results shown in Table 5.4, we have three major observations as follows. First, "GSM+THT" is superior to "GSM+noTHT", which means the improvement of semantic segmentation using THT versus no THT can still induce the better pose estimation performance. Second, [29] outperforms the average of "GSM+THT" because of its better semantic inference. However, it is still worse than "GSM+THT+Final" by more than 10% in both precision and recall, which further validates the entire incremental scene understanding framework. Third, we observe that the precision/recall

of the pose estimation is significantly higher than the semantic segmentation accuracies by around 15%/12%. This means that the our incremental pose estimation algorithm is able to compute correct poses with the partially correct segmentation.

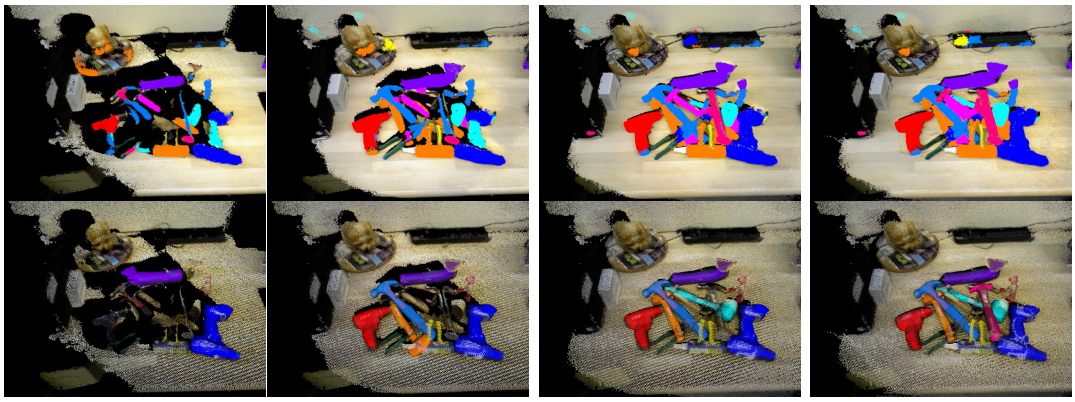
5.2.4.4 Qualitative Results and Runtime Analysis

Figure 5.9 shows some qualitative results of the semantic segmentation and pose estimation. Each row corresponds to a specific scene with objects that interact with each other and reside in complex background. We can see that our algorithm is capable of correcting wrongly predicted scene regions in previous frames at the end of each 400-frame sequence. More results can be found in the supplementary video.

Next, we analyze the runtime of our incremental semantic segmentation and pose estimation separately. Figure 5.8c shows the plot of average runtime at each frame. The incremental semantic segmentation takes nearly $3s$ at the first few frames and then its runtime rapidly drops below $1s$. In the last 100 frames, the processing time is below 0.5s. For the pose estimation, it is more fluctuated since ObjRecRANSAC needs to deployed on large areas with changed semantic labels sometimes. The average runtime of the semantic segmentation and pose estimation over all frames are 0.24s and 0.43s, respectively. Additionally, the GSM construction [22] runs at $4 \sim 5$Hz.

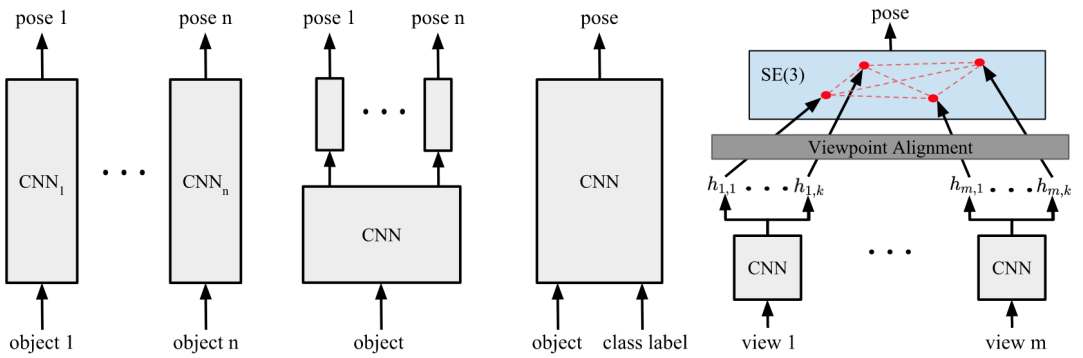


(a) Frame 50 **(b)** Frame 150 **(c)** Frame 250 **(d)** Frame 400



(e) Frame 50 **(f)** Frame 150 **(g)** Frame 250 **(h)** Frame 400

Figure 5.9: Example results of the semantic segmentation and pose estimation on the online reconstructed scenes are shown in upper and bottom parts in each subfigure, respectively. We show the results of two scenes at frame 50, 150, 250 and 400. Top and bottom rows correspond to Scene 18 and Scene 5. Each predicted semantic class and the associated estimated poses are highlighted by a unique and consistent color.



(a) Object Per Network (b) Object Per Output Branch (c) Multi-Class Network (d) Multi-View Multi-Class Framework

Figure 5.10: Illustration of different learning architectures for single-view object pose estimation: (a) each object is trained on an independent network; (b) each object is associated with one output branch of a common CNN basis; and (c) our network with single output stream via class prior fusion. Figure (d) illustrates our multi-view, multi-class pose estimation framework where $h_{m,k}$, the k-th pose hypothesis on view m, is first aligned to a canonical coordinate system and matched against other hypotheses for pose voting and selection.

5.3 End-to-End Learning for Multi-Class Pose Estimation

In this section, we first give an overview of our approach in Section 5.3.1. Subsequently, we introduce multi-class single-view network in Section 5.3.2 and the multi-view framework in Section 5.3.3. We evaluate our method in Section 5.3.4.

5.3.1 Introduction

Estimating 6-DoF object pose from images is a core problem for a wide range of applications including robotic manipulation, navigation, augmented reality and autonomous driving. While numerous methods appear in the literature [5, 16, 20, 54, 186, 64, 19, 63], scalability (to large numbers of objects) and

accuracy continue to be critical issues that limit existing methods. Recent work has attempted to leverage the power of deep CNNs to surmount these limitations [65, 66, 34, 67, 68, 18, 17, 69]. The simplest approach is to train a network for estimate the pose of each object of interest (Fig. 5.10 (a)). More recent approaches follow the principle of "object per output branch" (Fig. 5.10 (b)) whereby each object class is associated with an output stream connected to a shared feature basis [17, 18, 65, 66, 69]. In both cases, the size of the network increases with the number of objects which in turn implies that large amounts of data are needed for each class to avoid overfitting. In this work, we present a multi-class pose estimation architecture (Fig. 5.10 (c)) which receives object images and class labels provided by a detection system and which has a single branch for pose prediction. As a result, our model is readily scalable to large numbers of object categories and works for unseen instances while providing robust and accurate pose prediction for each object.

The ambiguity of object appearance and occlusion in cluttered scenes is another problem that limits the application of pose estimation in practice. One solution is to exploit additional views of the same instance to compensate for recognition failure from a single view. However, naive "averaging" of multiple single-view pose estimates in SE(3) [187] does not work due to its sensitivity to incorrect predictions. Additionally, most current approaches to multi-view 6-DoF pose estimation [104, 29, 105] do not address single-view ambiguities caused by object symmetry. This exacerbates the complexity of view fusion when multiple correct estimates from single views does not agree on SE(3). Motivated by these challenges, we demonstrate a new multi-view

framework (Fig. 5.10 (d)) which selects pose hypotheses, computed from our single-view multi-class network, based on a distance metric robust to object symmetry.

In summary, we make following contributions for scalable and accurate pose estimation on multiple classes and multiple views:

- We develop a multi-class CNN architecture for accurate pose estimation with three novel features: a) a single, non-branching generic pose representation which induces discriminative features across object categories; b) a method to embed object class labels into the learning process by concatenating a tiled class map with convolutional layers; and c) deep supervision with an object mask is performed so that we can exploit synthetic data to train models that generalize well to real images [30].

- We present a multi-view fusion framework which reduces single-view ambiguity with a novel voting scheme. An efficient implementation is proposed to enable fast hypothesis selection during inference. We empirically validate that our multi-view algorithm consistently improves the single-view pose estimation performance.

- We show our method provides state-of-the-art performance on public benchmarks including YCB-Video [17], JHUScene-50 [29] for 6-DoF object pose estimation [17, 29], and ObjectNet-3D for large-scale viewpoint estimation [34]. Further, we present a detailed ablative study on all benchmarks to empirically validate the three innovations for single-view pose estimation network.

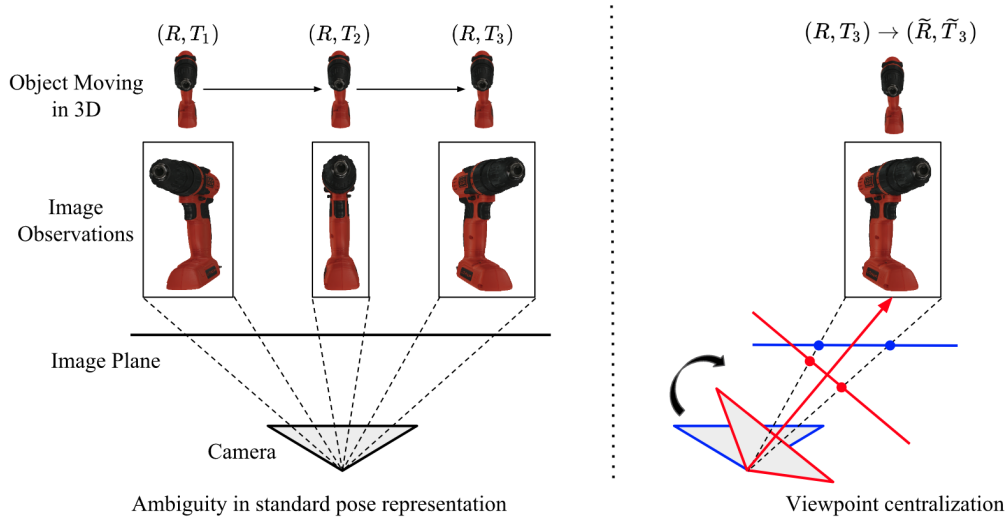


Figure 5.11: Illustration of the ambiguity in standard pose representation (left) and our solution of viewpoint centralization (right). The blue and red colors on the right figure indicate the camera and image planes before and after centralization, respectively.

5.3.2 Multi-Class Single-View Pose Estimation Network

In this section, we introduce a CNN-based architecture for multi-class pose estimation (Fig. 5.12 (c)). The input can be either cropped RGB or RGB-D object image provided by arbitrary detection algorithm. The network outputs are applied to compute both the rotation R and translation T of a 6-DoF pose (R, T). In the common practice of pose annotation process [5, 61, 29, 29, 17], we label R and T with respect to the current camera viewpoint by fitting the object model onto the observed 3D scene with the help of AR markers. Although the combined SE(3) representation (R, T) is consistent with the projected image appearance, the rotation component R can be ambiguous where the same rotation R corresponds to different object observation in image domain. We show a simple example in the left of Fig. 5.11. Here, a power drill with a

fixed rotation R is moving from left to right along the direction of X axis of image plane. If we capture its snapshot on the trajectory with different T, the corresponding image observations can be distinct as the drill undergoes out-of-plane rotation with respect to the camera viewpoint. Such inconsistency becomes a problem when we try to learn the generalizable mapping from image space to the rotation space (i.e. SO(3)). This issue has been revealed in the case of 1-D yaw angle estimation in [67]. Unfortunately, most existing learning-based 6-DoF pose estimation approaches [17, 18] ignore this problem where the cropped image or feature map directly regresses to R.

Our solution is to rectify the pose as the object is observed from the center-line of the camera. Consider the bounding box location of an observed object image I_o as (x_1, y_1, x_2, y_2) where (x_1, y_1) and (x_2, y_2) are image coordinates for top-left and bottom-right corners. Let (c_x, c_y) be the 2D camera center on image plane and f_x, f_y be the focal lengths for X and Y axes, respectively. We can compute the 3D orientation $\vec{v}$ towards the center of observed image I_o:

$$\vec{v} = [(\frac{x_1 + x_2}{2} - c_x)/f_x, (\frac{y_1 + y_2}{2} - c_y)/f_y, 1] \tag{5.14}$$

Subsequently, we compute rectified XYZ axes of camera coordinate system $[X_{\vec{v}}, Y_{\vec{v}}, Z_{\vec{v}}]$ by aligning the current Z axis $[0, 0, 1]$ to $\vec{v}$.

$$X_{\vec{v}} = [0, 1, 0] \times Z_{\vec{v}},\ Y_{\vec{v}} = Z_{\vec{v}} \times X_{\vec{v}},\ Z_{\vec{v}} = \frac{\vec{v}}{\|\vec{v}\|_2} \tag{5.15}$$

Note that symbol $\times$ is the cross product. Finally, the original pose label (R, T) can be projected onto this rectified XYZ axes to get the rectified pose $(\widetilde{R}, \widetilde{T})$:

$$\widetilde{R} = R_{\vec{v}} \cdot R, \quad \widetilde{T} = R_{\vec{v}} \cdot T, \tag{5.16}$$

where $R_{\vec{v}} = [X_{\vec{v}}; Y_{\vec{v}}; Z_{\vec{v}}]$ stacks the X, Y, Z coordinates in column. The right of Fig. 5.11 illustrates the process of viewpoint centralization. If the depth image and camera intrinsics are available, we also rectify the XYZ value of each image pixel by transforming each XYZ by $R_{\vec{v}}$. Subsequently, we construct a normalized XYZ map by centering the point cloud to its median.

In Fig. 5.11, we can see that RGB image plane is also rotated after the centralization. Therefore, the original object image is supposed to be warped to the new image plane in principle. However, this warping operation only changes the image scale but not the content (there is no out-of-plane rotation), which affects little on the CNN training because CNN always requires some types of scale normalization[7]. As a consequence, we leave the original image static while rectifying its pose label.

Figure 5.12 illustrates the details of our network design. Two streams of convolutional layers receive RGB image and XYZ map respectively and the final outputs are bin and delta vectors for both rotation and translation (Section 5.3.2.1). These two streams are further merged with class priors (Section5.3.2.2) and deeply supervised by object mask (Section 5.3.2.3).

5.3.2.1 Bin & Delta Representation for SE(3)

Direct regression to object rotation by L2 loss has been shown to be inferior to a classification scheme over a discretized SO(3) [48, 67, 18]. The common splitting strategy of SO(3) is to slice multiple bins along each dimension of Euler angle (α, β, γ) (i.e. yaw, pitch and roll) and supervise each discretized

[7]For example, isotropic and non-isotropic warping.

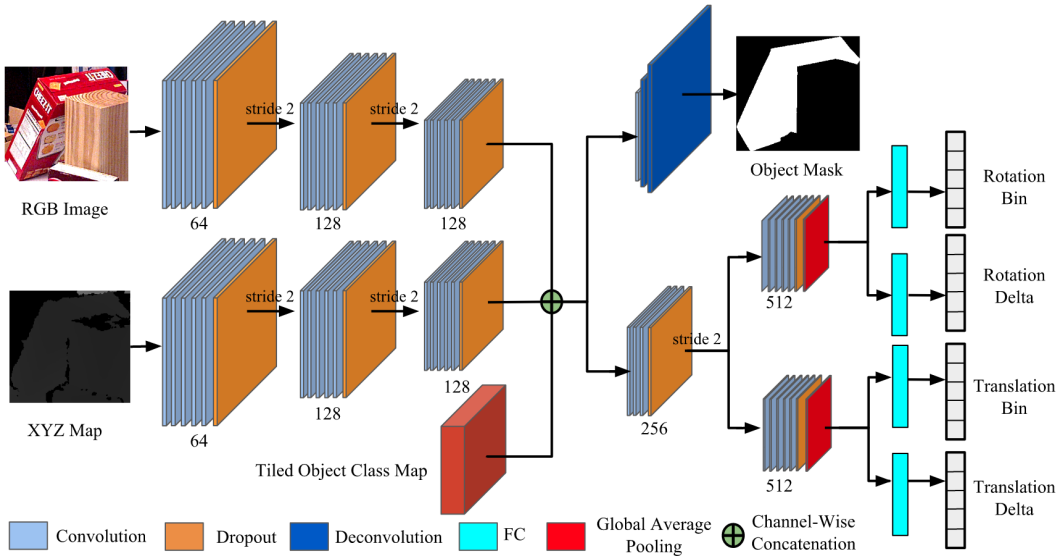


Figure 5.12: Multi-class network architecture on a single view. XYZ map stores normalized 3D coordinates of each pixel. If depth value is not available, we only train the stream of color image. The number of layers shown above is actually applied in our implementation.

dimension independently [65, 18]. However, this binning scheme yields a non-uniform tessellation of SO(3). Consequently, a small error on one Euler angle may be magnified to contribute large deviation in the final rotation estimate. To see this, consider a rotation $R = R_\gamma R_\beta R_\alpha$ composed by a sequence of rotations based on its Euler angles. The small error δ in predicting α can lead to large error in final prediction: $d(R_\gamma R_\beta R_\alpha, R_\gamma R_\beta R_{\alpha+\delta})$, where $d(R_1, R_2) = \frac{1}{2}\| \log(R_1^T R_2) \|_F$ is the geodesic distance between two rotations R_1 and R_2. In other words, the neighborhood of each bin center is not smooth and highly non-linear. In the following, we formulate two new bin & delta representations which uniformly partition both SO(3) and translation space. They are further coupled with classification & regression scheme for learning discriminative pose feature.

Almost Uniform Partition of SO(3) We first exploit the sampling technique developed by [188] to generate N rotations $\{\hat{R}_1, ..., \hat{R}_N\}$ that are uniformly distributed on SO(3). These N rotations are used as the centers of N rotation bins in SO(3). Given an arbitrary rotation matrix R, we convert it to a pair of bin and delta representation $(\vec{b}^R, \vec{d}^R)$ based on $\{\hat{R}_1, ..., \hat{R}_N\}$. Bin vector $\vec{b}^R$ contains N dimensions where the i-th dimension $\vec{b}_i^R$ indicates the confidence of R belonging to bin i. $\vec{d}^R$ stores N rotations (i.e. quaternions in our implementation) where the i-th rotation $\vec{d}_i^R$ is the deviation from $\hat{R}_i$ to R. We note that $\{\hat{R}_1, ..., \hat{R}_N\}$ is shared between multiple objects because it is a generic set of bin centers that uniformly covers SO(3) regardless of object classes.

Next, we enforce a sparse confidence scoring scheme for $(\vec{b}^R, \vec{d}^R)$. Given a rotation R, we only activate a subset of representative bins and deltas. Formally, we compute $\vec{b}_i^R$ and $\vec{d}_i^R$ as follows:

$$\vec{b}_i^R = \begin{cases} \theta_1 & : i \in NN_1(R) \\ \theta_2 & : i \in NN_k(R) \setminus NN_1(R) \\ 0 & : \text{Otherwise} \end{cases}, \quad \vec{d}_i^R = \begin{cases} R \cdot \hat{R}_1^T & : i \in NN_k(R) \\ 0 & : \text{Otherwise} \end{cases} \tag{5.17}$$

where $NN_k(R)$ is the set of k nearest neighbors of R among $\{\hat{R}_1, ..., \hat{R}_N\}$ in terms of the geodesic distance $d(R_1, R_2) = \frac{1}{2}\|\log(R_1^T R_2)\|_F$ between two rotations R_1 and R_2. In principle, θ_1 should be significantly larger than θ_2. Note that we design delta $\vec{d}_i$ to achieve $R = \vec{d}_i^R \cdot \hat{R}_i^T$ and not $R = \hat{R}_i^T \cdot \vec{d}_i^R$. For the second case, small prediction error on $\vec{d}_i^R$ may cause large error on final prediction of R even if the bin prediction is correct. During inference, we take the bin with maximum score and apply the corresponding delta value to the

bin center to compute the final prediction.

Gridding XYZ Axes. We represent translations by uniformly gridding X, Y and Z axes separately. For RGB-D data, the XYZ axes are defined to be the coordinate axes of normalized point cloud (i.e. XYZ map). The translation vector is the spatial deviation from the origin to the 3D object center. For a cropped RGB image with known camera intrinsics, we set X and Y axes as image coordinates and Z axis as the viewing ray of the camera. Therefore, X and Y coordinates indicate the image location of projected 3D object center and Z value remains as the depth distance. Because we conduct the non-isotropic warping to resize an RGB image to a normalized network input with a fixed scale, we further adjust Z to Z' such that image scale is consistent to depth value: $Z' = Z \cdot \frac{s'}{s}$, where s' and s are image scales before and after resizing, respectively.

Here we discuss how to construct the bin & delta pair $(\vec{b}^{T_x}, \vec{d}^{T_x})$ for X axis. Y and Z axes are done in the same way. We slice M non-overlapping bins with equal size $\frac{s_{max}-s_{min}}{M}$ between $[s_{min}, s_{max}]$ [8]. When X value is lower than s_{min} (or larger than s_{max}), we assign it to the first (or last bin). Similar to Equation 5.17, we compute $\vec{b}^{T_x}$ of an X value by finding its K' nearest neighbors among M bins on X axis and assigning θ'_1 for the top nearest neighbor as well as θ'_2 for the remaining $K-1$ neighbors ($\theta'_1 \gg \theta'_2$). Correspondingly, the delta values of the K' nearest neighbor bins are deviations from the bin centers to the actual X value and others are 0. Similar to SO(3), we compute X value during inference by adding the delta to the bin center which achieves the

[8] s_{min} and s_{max} may vary across different axes

maximum confidence score. Finally, we concatenate all bins and deltas of X, Y and Z axes: $\vec{b}^T = [\vec{b}^{T_x}, \vec{b}^{T_y}, \vec{b}^{T_z}]$ and $\vec{d}^T = [\vec{d}^{T_x}, \vec{d}^{T_y}, \vec{d}^{T_z}]$. One alternative way of dividing translation space is to apply joint griding over XYZ space. However, the total number of bins grows exponentially as M increases and we found no performance gain by doing so in practice.

5.3.2.2 Fusion of Class Prior

Many existing methods assume known object class labels, provided by a detection system, prior to pose analysis [17, 18, 48, 66, 20]. However, they ignore the class prior during training and only apply it for inference purpose. Our idea is to seamlessly fuse this known class label into the learning process of convolutional filters. This is partly inspired by CNN-based hand-eye coordination learning [189] where a tiled robot motor motion map is concatenated with one hidden convolutional layer for predicting the grasp success probability. Given the class label of the crop image, we create a one-hot vector where the entry corresponding to the class label is set to 1 and all others to 0. We further spatially tile this one-hot vector to form a 3D tensor with size $H \times W \times C$, where C is the number of object classes and H, W are height and width of a convolutional feature map at an intermediate layer. As shown in Fig. 5.12, we concatenate this tiled class tensor with the last convolutional layers of both color and depth streams along the filter channel. Therefore, the original feature map is embed with class labels at all spatial locations and the following layers are able to model class-specific patterns for pose estimation. This is critical in teaching the network to develop compact class-specific filters for each individual object while taking advantage of a shared basis of low

level features for robustness.

5.3.2.3 Deep Supervision with Object Segmentation

Due to scarce pose annotations on real images, synthetic CAD renderings are commonly used as training data for learning-based pose estimation methods [17, 5, 18]. Inspired by [30], we incorporate a deep supervision module into our multi-class pose network for additional regularization. Besides the object class label, the multi-class network should be capable of segmenting an object from cluttered background for correct pose inference. We can view the object mask as an "intermediate" concept for the final task of 6-DoF pose estimation. That is, good object segmentation is a prerequisite for the final success of pose estimation. Moreover, precisely predicted object mask benefits some post-refinement steps such as Iterative Closest Point (ICP). We impose the deep supervision of object mask at a hidden layer, as shown in Fig. 5.12. After the combination of feature and class maps (Section 5.3.2.2), we append one output branch for object mask which contains one convolution layer followed by two de-convolution layers with upsampling ratio 2. The object mask is a binary map where "1" indicates object pixel and "0" means background or other objects.

5.3.2.4 Network Architecture

Putting this all together, the overall loss function consists of five loss components over the segmentation map, the rotation, and three translation components:

$$\mathcal{L} = l_{seg} + l_{R_b}(\widetilde{\vec{b}^R}, \vec{b}^R) + l_{R_d}(\widetilde{\vec{d}^R}, \vec{d}^R) + \sum_{i \in \{X,Y,Z\}} \left(l_{T_b}(\widetilde{\vec{b}^{T_i}}, \vec{b}^{T_i}) + l_{T_d}(\widetilde{\vec{d}^{T_i}}, \vec{d}^{T_i}) \right) \tag{5.18}$$

where $\widetilde{\vec{b}^R}$, $\widetilde{\vec{d}^R}$, $\widetilde{\vec{b}^{T_i}}$ and $\widetilde{\vec{d}^{T_i}}$ are the bin and delta estimates of the groundtruth $\vec{b}^R$, $\vec{d}^R$, $\vec{b}^{T_i}$ and $\vec{d}^{T_i}$, respectively. We apply cross-entropy softmax to segmentation loss l_{seg} on each pixel location, SO(3) bin loss l_{R_b} and translation bin loss l_{T_b}. In addition, we use L2 loss for the delta losses l_{R_d} and l_{T_d}. All losses are simultaneously backpropagated to the network to update network parameters on each batch. For simplicity, we apply loss weight 1 for each loss function.

In our network, each convolutional layer is coupled with a batch-norm layer [44] and ReLU. The size of all convolutional filters is 3x3. The output layer for each bin and delta is constructed with one global average pooling (GAP) layer followed by one fully connected (FC) layer with 512 neurons. We employ dropout [9] layer before each downsampling of convolution with stride 2. We deploy 23 layers in total.

5.3.3 Multi-View Pose Hypothesis Selection

In this section, we present a multi-view framework which refines the outputs of our single-view network (Section 5.3.2) during an inference stage. We assume that camera pose of each frame in a sequence is known. In practice,

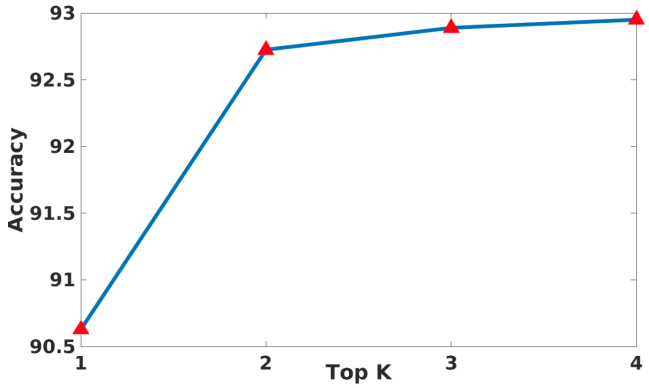


Figure 5.13: Top-K accuracies of our single-view pose network over all object classes in YCB-Video benchmark [17]. We use RGB-D data as network input.

this framework can be connected to many scalable and precise SLAM systems such as [21].

5.3.3.1 Motivation

Recall that the single-view pose network predicts confidence scores of all bins in SO(3), X, Y, and Z spaces (Section 5.3.2.1). Therefore, we are able to extract top-K estimates from each space which achieve the K highest confidence scores. Subsequently, we can compute K^4 pose hypotheses by composing top-k results from all spaces.

To evaluate the quality of these hypotheses, we compute top-K accuracies where the best hypothesis that achieves the lowest pose error is selected as the final prediction result. Fig. 5.13 shows the curve of top-K accuracies across all object classes, in terms of mPCK on YCB-Video benchmark [17]. Please refer to Section 5.3.4 for more details on mPCK metric. We observe that the pose estimation performance significantly improves when we initially increase K value from 1 to 2 and almost saturates when K proceeds to 4. This

result indicates that the inferred confidence score is ambiguous up to a small range, which makes sense especially for objects with symmetrical geometry or texture. Then the question is how we can resolve this ambiguity and further improve the pose estimation performance. Next, we present a multi-view voting algorithm to select correct hypothesis from the top-K hypothesis set.

5.3.3.2 Hypothesis Voting

We consider a hypothesis set $\mathcal{H} = \{h_{1,1}, \cdots, h_{i,j}, \cdots, h_{n,K^4}\}$ from n views, where $h_{i,j}$ indicates the hypothesis j in view i and K^4 pose hypotheses on each view. To measure the difference between hypotheses from different views, we need to first transform each $h_{i,j} \in \mathcal{H}$ into the same coordinate frame. For simplicity, we register all hypotheses into the camera coordinate of view 1 given the camera poses of all n views. We denote $\mathcal{T}_i$ as the transformation for view i so $\mathcal{T}_1$ is an identity transformation. Next, we compute the pairwise distance $D(h_{i,j}, h_{p,q})$ for every two hypotheses $h_{i,j}, h_{p,q} \in \mathcal{H}$. The voting score $V(h_{i,j})$ for $h_{i,j}$ is then calculated as follows:

$$V(h_{i,j}) = \sum_{h_{p,q} \in \mathcal{H} \setminus h_{i,j}} \max\Big(\sigma - D(\mathcal{T}_i(h_{i,j}), \mathcal{T}_p(h_{p,q})), 0\Big) \tag{5.19}$$

where σ is a pre-defined threshold for outlier rejection. We select the hypothesis with the highest vote score as the final prediction. To handle single-view ambiguity caused by symmetrical geometry, we follow the same distance metric proposed by [5] to measure the discrepancy between two hypothesis

$h_1 = (R_1, T_1)$ and $h_2 = (R_2, T_2)$:

$$D(h_1, h_2) = \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|(R_1 x_1 + T_1) - (R_2 x_2 + T_2)\|_2 \tag{5.20}$$

where $\mathcal{M}$ denotes the set of 3D model points and $m = |\mathcal{M}|$. Note that $D(h_1, h_2)$ yields small distance when 3D object occupancies computed by h_1 and h_2 are similar, even if h_1 and h_2 have large geodesic distance on SO(3). This entire multi-view voting process is illustrated in Fig. 5.10 (d).

One alternative way to combine multiple hypotheses is to simply "max" or "average" pool them. For "max" pooling, we compute the final estimate by using the bin and delta with highest confidence score, which fails to the exploit the top-K results and is essentially the same as in single-view case. In addition, the "average" of multiple SE(3) estimates can be performed by principled approaches such as [187]. However, it is sensitive to incorrect predictions or outliers.

5.3.3.3 Efficient Implementation

The above hypothesis voting algorithm is computationally expensive because the time complexity of Equation 5.20 is at least $O(m \log m)$ via a KDTree implementation. Our solution is to decouple translation and rotation components in Equation 5.20 and approximate $D(h_1, h_2)$ by $\widetilde{D}(h_1, h_2)$:

$$\widetilde{D}(h_1, h_2) = \|T_1 - T_2\|_2 + \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R_1 x_1 - R_2 x_2\|_2 \tag{5.21}$$

In fact, $\widetilde{D}(h_1, h_2)$ is an upperbound of $D(h_1, h_2)$: $D(h_1, h_2) \leq \widetilde{D}(h_1, h_2)$ for any h_1 and h_2, because $\|(R_1 x_1 + T_1) - (R_2 x_2 + T_2)\|_2 \leq \|R_1 x_1 - R_2 x_2\| +$

$\|T_1 - T_2\|$ based on the triangle inequality. We can see that the complexity of calculating $\|T_1 - T_2\|$ is $O(1)$. Therefore, we focus to speed up the computation of rotation distance $\frac{1}{m}\sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R_1 x_1 - R_2 x_2\|_2$. The idea is to pre-compute a table of this pairwise distance between every two rotations among N pre-defined rotations $\{\hat{R}_1, ..., \hat{R}_N\}$. $\{\hat{R}_1, ..., \hat{R}_N\}$, computed by the same uniform sampling technique [188] as used in Section 5.3.2.1, forms a uniform and dense coverage over SO(3). For arbitrary R_1 and R_2, we search for their nearest neighbors $\hat{R}_{N_1(R_1)}$ and $\hat{R}_{N_1(R_2)}$ from $\{\hat{R}_1, ..., \hat{R}_N\}$. In turn, we approximate the rotation distance as follows:

$$\frac{1}{m}\sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R_1 x_1 - R_2 x_2\|_2 \approx \frac{1}{m}\sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|\hat{R}_{N_1(R_1)} x_1 - \hat{R}_{N_1(R_2)} x_2\|_2 \tag{5.22}$$

where the right hand side can be directly retrieved from the pre-computed distance table during inference. When N is large enough, the approximation error of Equation 5.22 affects little on our voting algorithm. In practice, we find the performance gain saturates when $N \geq 1000$. Thus, the complexity of Equation 5.22 is $O(\log N)$ for nearest neighbor search, which is significantly smaller than $O(m \log m)$ of Equation 5.19 ($m > N$ in general). As a consequence, this efficient algorithm is capable of significantly speeding up the multi-view voting scheme.

5.3.4 Experiment

In this section, we empirically evaluate our method on large-scale datasets: YCB-Video [17], JHUScene-50 [29] for 6-DoF pose estimation, and ObjectNet-3D [34] for viewpoint estimation. Further, we conduct an ablative study to

validate our three innovations for single-view multi-class pose network.

Evaluation Metric. For 6-DoF pose estimation, we follow the recently proposed metric "ADD-S" by [17]. The traditional metric [5] considers a correct pose estimate h if $D(h, h^*)$ in Equation 5.20 is below a threshold with respect to its groundtruth h^*. [17] improves this threshold-based metric by computing the area under the curve of accuracy-threshold while varying different thresholds within a range (i.e. $[0, 0.1]$). We denote this new metric as "mPCK" because it is essentially the mean of PCK accuracy [31]. For viewpoint estimation, we use Average Viewpoint Precision (AVP) used in PASCAL3D+ [32] and Average Orientation Similarity (AOS) used in KITTI [172].

Implementation Details. The number of nearest neighbors we use for soft binning is 4 for SO(3) and 3 for each of XYZ axes. We set binning scores as $\theta_1 = \theta_1' = 0.7$ and $\theta_2 = \theta_2' = 0.1$. The number of rotation bins is 60. For XYZ binning, we use 10 bins and $[s_{min}, s_{max}] = [-0.2, 0.2]$ for each axis when RGB-D data is used. For inference on RGB data, we use 20 bins, $[s_{min}, s_{max}] = [0.2, 0.8]$ for XY axes and 40 bins, $[s_{min}, s_{max}] = [0.5, 4.0]$ for Z axis. In multi-view voting, we set the distance threshold $\sigma = 0.02$ and the precomputed size of distance table as 2700. The input image to our single-view pose network is 64x64. The tile class map is inserted at convolutional layer 15 with size $H = W = 16$. We use stochastic gradient descent with momentum 0.9 to train our network from scratch. The learning rate starts at 0.01 and decreases by one-tenth every 70000 steps. The batch size is 105 for YCB-Video (21 classes) and 100 for both JHUScene-50 (10 classes) and ObjectNet-3D (100 classes). We construct each batch by mixing equal number of data from each class. We name our

Object	RGB			RGB-D				
	P-CNN [17]	MCN	MV5-MCN	P-CNN + ICP [17]	MCN	MCN + ICP	MV5-MCN	MV5-MCN + ICP
002_chef_can	84.4	87.8	**90.6**	95.7	89.4	96.0	96.2	**97.8**
003_cracker_box	**80.8**	64.3	72.0	**94.8**	85.4	88.7	90.9	91.3
004_sugar_box	77.5	82.4	**87.4**	**97.9**	92.7	97.3	95.3	95.5
005_tomato_can	85.3	87.9	**91.8**	95.0	93.2	96.5	97.5	**98.0**
006_mustard_bottle	90.2	92.5	**94.3**	**98.2**	96.7	97.7	97.0	97.8
007_tuna_fish_can	81.8	84.7	**89.6**	96.2	95.1	97.6	95.1	**98.1**
008_pudding_box	**86.6**	51.0	51.7	**98.1**	91.6	86.2	94.5	95.2
009_gelatin_can	86.7	86.4	**88.5**	**98.9**	94.6	97.6	96.0	97.9
010_meat_can	78.8	83.1	**90.3**	84.0	91.7	90.8	96.7	**97.1**
011_banana	80.8	79.1	**85.0**	96.5	93.8	**97.5**	96.2	**97.5**
019_pitcher_base	81.0	84.8	**86.1**	97.4	93.8	96.6	96.2	**96.9**
021_cleanser	75.7	76.0	**81.0**	89.2	92.9	96.4	95.4	**97.8**
024_bowl	74.2	76.1	**80.2**	**91.7**	82.6	76.0	82.0	79.9
025_mug	70.0	91.4	**93.1**	94.2	95.3	97.3	96.8	**98.2**
035_power_drill	73.9	76.0	**81.1**	**98.0**	88.2	95.9	93.1	96.6
036_wood_block	**63.9**	54.0	58.4	93.1	81.5	93.5	93.6	**93.9**
037_scissors	57.8	71.6	**82.7**	94.6	87.3	79.2	94.2	**94.7**
040_large_marker	56.2	60.1	**66.3**	97.8	90.2	**98.0**	95.4	97.7
051_large_clamp	34.3	66.8	**77.5**	81.5	91.5	94.0	93.3	**95.4**
052_larger_clamp	38.6	61.1	**68.0**	51.6	88.0	90.7	90.9	**91.8**
061_foam_brick	**82.0**	60.9	67.7	96.4	93.2	96.5	95.9	**98.0**
All	73.4	75.1	**80.2**	93.1	90.6	93.3	94.3	**95.6**

Table 5.6: mPCK accuracies achieved by different methods on YCB-Video dataset [17]. The last row indicates the average-per-class of mPCKs of all classes.

Multi-Class pose Network as "MCN". The multi-view framework using n views is called as "MVn-MCN".

5.3.4.1 YCB-Video

YCB-Video dataset [17] contains 92 real video sequences. 80 videos along with 80,000 synthetic images are used for training and 2949 key frames are extracted from the remaining 12 videos for testing. We finetune the current state-of-the-art "mask-RCNN" [49] on the training set as the detection system. Following the same scenario in [17], we assume that one object appears at

Object	mPCK on GT		Segmentation Accuracy	
	RGB	RGB-D	RGB	RGB-D
002_chef_can	91.2	94.4	92.5	94.5
003_cracker_box	78.5	86.3	88.8	89.2
004_sugar_box	85.1	93.8	91.8	94.5
005_tomato_can	93.3	94.2	90.0	94.6
006_mustard_bottle	91.9	96.8	97.3	97.5
007_tuna_fish_can	95.2	95.8	90.2	93.5
008_pudding_box	84.9	90.9	69.3	61.3
009_gelatin_can	92.1	95.6	90.7	92.9
010_meat_can	90.8	87.0	87.7	88.8
011_banana	70.0	94.9	95.7	96.3
019_pitcher_base	91.1	94.6	94.2	96.0
021_cleanser	86.8	94.4	94.8	96.8
024_bowl	85.0	83.1	93.5	85.8
025_mug	91.9	95.5	89.5	87.9
035_power_drill	87.2	91.3	85.4	89.9
036_wood_block	87.2	83.7	83.5	89.1
037_scissors	80.2	75.0	92.8	92.5
040_large_marker	66.4	89.2	89.0	93.4
051_large_clamp	86.5	92.7	88.0	90.4
052_larger_clamp	79.5	87.5	92.1	92.9
061_foam_brick	79.2	93.9	90.6	91.5
All	86.4	91.0	89.9	90.9

Table 5.7: mPCK and instance segmentation accuracies of MCN on YCB-Video Dataset.

most once in a scene. Therefore, we compute the bounding box of a particular object by finding the one with highest detection score of that object. For our multi-view system, one view is coupled with 5 other randomly sampled views in the same sequence. Each view outputs top-3 results from each space of SO(3), X, Y and Z and in turn $3^4 = 81$ pose hypotheses.

Table 5.6 reports mPCK accuracies of our methods and variants of poseCNN [17] (denoted as "P-CNN"). We first observe that the multi-view framework (MV5-MCN) consistently improves the single-view network (MCN) across

different classes and achieves the overall state-of-the-art performance. Such improvement is more significant on RGB data, where the mPCK margin between MV5-MCN and MCN is 5.1% which is much larger than the margin of 1.0% on RGB-D data for all classes. This is mainly because single-view ambiguity is more severe without depth data. Subsequently, MCN outperforms poseCNN by 1.7% on RGB and MCN+ICP is marginally better than poseCNN+ICP by 0.2% on RGB-D. We can see that MCN achieves more balanced performance than poseCNN across different classes. For example, poseCNN+ICP only obtains 51.6% on class "052_larger_clamp" which is 24.4% lower than the minimum accuracy of a single class by MCN+ICP. This can be mainly attributed to our class fusion design in learning discriminative class-specific feature so that similar objects can be well-separated in feature space (e.g. "051_large_clamp" and "052_larger_clamp"). Finally, post refinement technique ICP is able to further improve the pose estimation results from MCN or MV5-MCN when depth data is available.

Table 5.7 reports the mPCK accuracies of all object classes on groundtruth (GT) of object bounding boxes. We can see that the overall mPCK on RGB is 86.4% on RGB (11.8% higher than the result on RGB) and mPCK on RGB-D is 91.0% (0.4% higher than the result on RGB-D). Typically, the gap of mPCK between GT and detection is larger on RGB than RGB-D. This is because we rely on actual image scale of bounding box to recover 3D translation for RGB input. Consequently, the translation estimate is sensitive to the jittering of object bounding boxes. Additionally, Table 5.7 also reports the segmentation accuracies of MCN on both RGB and RGB-D. It is a little bit surprising that

Object	RGB			RGB-D			
	PM [20]	MCN	MV5-MCN	ORR [4]	PM [20]	MCN	MV5-MCN
drill_1	10.6	33.4	**36.5**	14.5	70.3	76.8	**78.1**
drill_2	9.9	48.8	**54.5**	2.9	49.0	76.6	**80.1**
drill_3	7.6	45.5	**48.0**	3.7	50.9	81.5	**85.4**
drill_4	9.3	41.6	**45.5**	6.5	51.4	82.0	**87.1**
hammer_1	5.0	24.9	**30.2**	8.1	38.7	80.1	**87.6**
hammer_2	5.1	28.3	**33.4**	10.7	35.5	81.2	**91.5**
hammer_3	7.8	26.2	**31.2**	8.6	47.8	83.1	**88.1**
hammer_4	5.1	17.2	**20.6**	3.8	38.3	73.8	**87.8**
hammer_5	5.2	37.1	**44.4**	9.6	35.0	78.0	**86.3**
sander	10.7	35.6	**39.5**	9.5	54.3	**76.0**	75.5
All	7.6	33.9	**38.4**	7.8	47.1	78.9	**84.8**

Table 5.8: mPCK accuracies of all objects in JHUScene-50 dataset [29]. The last row indicates the average-per-class of mPCKs of all classes. Best results are highlighted in bold. "PM" is short for the Pose Manifold learning method [20]. "ORR" stands for ObjRecRANSAC [4].

the segmentation performance on RGB only is quite competitive to the one on RGB-D. We obtain high instance segmentation accuracy[9] of MCN across all classes: 89.9% on RGB and 90.9% on RGB-D. This implies that MCN does actually learn the intermediate foreground concept for final pose prediction. The mPCK on RGB is even higher than the one on RGB-D on some classes such as "008_pudding_box". This implies that RGB images offer the critical details for instance segmentation.

5.3.4.2 JHUScene-50

JHUScene-50 [29] contains 50 scenes with diverse background clutter and heavy object occlusion. Moreover, the target object set (10 hand tools) consists

[9]The ratio of the number of pixels with correctly predicted mask label versus all

of many instances with similar appearances. Only textured CAD models are available during training and all 5000 real image frames construct the test set. To cope with our pose learning framework, we simulate a large amount of synthetic data by rendering densely cluttered scenes similar to test data, where objects are randomly piled on a table. We use Unreal Engine[10] as the rendering engine and generate 100k training images.

We compare MCN and MV5-MCN with ObjRecRANSAC[11] [4] in JHUScene-50 and one recent state-of-the-art pose manifold learning technique [20][12]. We compute 3D translation for [20] by following the same procedure used in [5]. We evaluate different methods on the groundtruth locations of all objects. Table 5.8 reports mPCK accuracies of all methods. We can see that MCN significantly outperforms other comparative methods by great margins, though MCN performs much worse than on YCB-Video mainly because of more severe occlusion and diverse cluttered background in JHUScene-50. Additionally, we observe that MV5-MCN is superior to MCN on both RGB and RGB-D data. The performance gain on RGB-D data achieved by MV5-MCN is much larger than the one on YCB-Video, especially for the hammer category due to the symmetrical 3D geometry.

Table 5.9 shows the segmentation accuracies of MCN on RGB and RGB-D data. We observe the similar phenomenon that the segmentation accuracy on RGB is only slightly worse than the result on RGB-D. Additionally, the overall segmentation accuracy on JHUScene-50 is roughly 10% lower than

[10]https://www.unrealengine.com/en-US/what-is-unreal-engine-4
[11]https://github.com/tum-mvp/ObjRecRANSAC
[12]We re-implement this method because the source code is not publicly available.

Object	RGB	RGB-D
drill_1	73.7	74.7
drill_2	69.6	73.3
drill_3	75.0	79.4
drill_4	69.2	76.0
hammer_1	88.2	89.5
hammer_2	87.1	88.9
hammer_3	81.7	83.3
hammer_4	87.7	88.5
hammer_5	88.4	88.6
sander	79.4	79.1
All	80.0	82.1

Table 5.9: Instance segmentation accuracies of MCN on JHUScene-50 dataset [29].

mAP	AOS		AVP	
Fast R-CNN [47]	ObjectNet-3D [34]	MCN	ObjectNet-3D [34]	MCN
61.6	51.9	**56.0**	39.4 (64.0)	**50.0** (**81.2**)

Table 5.10: Accuracies of object pose estimation on ObjectNet-3D benchmark [34]. All methods perform over the same set of detected bounding boxes estimated by Fast R-CNN [47]. Best results on both AOS and AVP metrics are shown in bold. For AVP, we also report $\frac{\text{AVP}}{\text{mAP}}$ in parentheses.

the one on YCB-Video, which is consistent with the fact that JHUScene-50 is inferior to YCB-Video in terms of mPCK accuracy achieved by MCN. This is mainly because MCN is trained on synthetic images only on JHUScene-50 while a mixture of synthetic and real training images being used in YCB-Video. Further, the real test data of JHUScene-50 contains heavier occlusion and more diverse cluttered background.

5.3.4.3 ObjectNet-3D

To evaluate the scalability of our method, we conduct the experiment on ObjectNet-3D which consists viewpoint annotation of 201,888 instances from 100 object categories. In contrast to most existing benchmarks [17, 29, 5] which target for indoor scenes and small objects, ObjectNet-3D covers a wide range of outdoor environments and diverse object classes such as aeroplane. We modify MCN model by only using the rotation branch for viewpoint estimation and removing the deep supervision of object mask because object mask is not available in ObjectNet-3D. To our knowledge, only [34] reports viewpoint estimation accuracy on this dataset, where a viewpoint regression branch is added along with bounding box regression in Fast R-CNN architecture [47]. For the fair comparison, we use the same detection results for [34] as the input to MCN. Because ObjectNet-3D only provides detection results on the validation set, we train our model on the training split and test on the validation set. Table 5.10 reports the viewpoint estimation performance on two different metrics AVP [32] and AOS [172]. The detection performance in mAP is the upperbound of AVP. The numbers in parentheses are the ratios of AVP versus mAP. We can see that MCN is significantly superior to the large-scale model [34] on both AOS and AVP, even if [34] actually optimizes the network hyper-parameters on the validation set. This shows that MCN can be scaled to a large-scale pose estimation problem. Moreover, object instances have little overlap between training and validation sets in ObjectNet-3D, which indicates that MCN is capable of generalizing to unseen object instances within a category.

Method	RGB			RGB-D	
	YCB-Video	JHU	ObjectNet-3D	YCB-Video	JHU
plain	61.0	25.0	51.7 / 38.3	61.8	19.6
no tiled class	66.2	26.3	50.3* / 41.3*	89.5	70.0
no segmentation	68.5	29.3	**56.0** / **50.0**	90.1	76.4
Sep. branch + Seg. + BD	73.8	31.6	52.5* / 42.9*	90.2	77.7
Sep. network + Seg. + BD	62.1	28.7	NA	87.1	66.9
MCN (seg. + tiled class + BD)	**80.2**	**33.9**	NA	**90.8**	**78.9**

Table 5.11: An ablative study of different variants of pose estimation architectures on YCB-Video, JHUScene-50 and ObjectNet-3D. We follow the same metrics as we evaluate in previous sections. For ObjectNet-3D, we report accuracies formatted as AOS / AVP. The "*" symbol indicates that no segmentation mask is used in training because it is unavailable in ObjectNet-3D.

5.3.4.4 Ablative Study

In this section, we empirically validate the three innovations introduced for MCN: bin & delta representation ("BD"), tiled class map and deep supervision of object segmentation ("Seg."). Additionally, we also inspect the baseline architectures: separate network for each object ("Sep. network") and separate output branch for each object ("Sep. branch"), as shown in Fig. 5.10 (a) and Fig. 5.10 (b) respectively. To remove the effect of using "BD", we directly regress quaternion and translation (plain) as the comparison. Table 5.11 presents accuracies of different methods on all three benchmarks. We follow previous sections to report mPCK for YCB-Video and JHUScene-50, and AOS/AVP for ObjectNet-3D. Because ObjectNet-3D does not provide segmentation groundtruth, we remove module "Seg." in all analysis related

to ObjectNet-3D. Also, we do not report accuracy of "Sep. network" on ObjectNet-3D because it requires 100 GPUs for training. We have three main observations: 1. By removing any of three innovations, the pose estimation performance consistently decreases. Typically, "BD" is a more critical design than "Seg." and tiled class map because the removal of BD causes larger performance drop; 2. "Sep. branch" coupled with "BD" and "Seg." appears to be the second best architecture, but it is still inferior to MCN especially on YCB-Video and ObjectNet-3D. Moreover, the model size of "Sep. branch" grows rapidly with the increasing number of classes; 3. "Sep. network" is expensive in training and it significantly performs worse than MCN mainly because MCN exploits more diverse data from different classes to reduce overfitting.

5.3.4.5 Texture-Sensitive Metric

One of remaining questions is how good the pose estimation results by MCN preserves both the geometry and texture of the actual groundtruth. To answer this question, we need to use a different distance metric rather than the default one (Equation 5.20) "ADD-S [17]" to measure the distance between two poses. To this extent, we follow [17] to adopt "ADD" metric which defines a correct pose estimation as the one that not only matches the groudtruth pose in 3D geometry but also aligns the texture of the target. Therefore, "ADD" is a more strict metric than "ADD-S" in the sense that it requires a pose estimation system to yield exactly the same pose as groundtruth in SE(3). We report the overall mPCK accuracies of both "ADD" and "ADD-S" on YCB-Video and JHUScene-50 in Table 5.12. We observe two main results. First, "ADD"

Method	RGB		RGB-D	
	ADD-S	ADD	ADD-S	ADD
YCB-Video	80.2	63.3	90.8	69.6
JHU	33.9	15.5	78.9	36.7

Table 5.12: mPCK accuracies on all object classes of MCN on YCB-Video and JHUScene-50 datasets.

accuracies are in general much smaller than "ADD-S", which indicates that our current pose estimation system is limited in predicting accurate estimate in SE(3) space. This may be attributed to our low-resolution input size (64x64) so that MCN is hard to exploit texture details for precise estimation. Second, the performance drop on RGB-D data is more than the one on RGB data. This fact makes sense because RGB image is the dominant cue in resolving the geometry ambiguity and predicting actual pose in SE(3).

5.3.4.6 Qualitative Analysis

We visualize pose estimation results on YCB-Video in Fig. 5.14 and JHUScene-50 in Fig. 5.15. We can see that MCN is capable of predicting object pose under occlusion and further refines the MCN result. The multi-view algorithm MV5-MCN is capable of further boosting the result of MCN on RGB-D data, especially for objects with symmetrical geometry such as cup, bottle and bowl.

5.4 Conclusion

In this chapter, we first present how the proposed semantic segmentation algorithm in Section 3.5 boosts the performance of an off-the-shelf object pose registration technique [4] based on geometry matching. Subsequently, we

Figure 5.14: Illustration of pose estimation results by MCN on YCB-Video. The projected object mesh points that are transformed by pose estimates are highlighted by orange. From left to right of each data, we show original image, MCN estimates on RGB, MCN estimates on RGB-D and MV5-MCN estimates on RGB-D.

Figure 5.15: Illustration of pose estimation results by MCN on JHUScene-50. The projected object mesh points that are transformed by pose estimates are highlighted by pink. From left to right of each data, we show original image, MCN estimates on RGB, MCN estimates on RGB-D and MV5-MCN estimates on RGB-D.

introduce a SLAM-enhanced incremental scene understanding framework which improves the single-view semantic segmentation in terms of both speed and accuracy. Finally, we present a unified learning architecture for inferring 6-DoF object pose from single and multiple views. We first introduce three innovations for deep CNNs: a new bin & delta pose representation, the fusion of tiled class map into convolutional layers and deep supervision of object mask at intermediate layer. These modules enable a scalable pose learning architecture for large-scale object classes and unconstrained background clutter. Subsequently, we formulate a new multi-view framework for selecting single-view pose hypotheses while considering ambiguity caused by object symmetry. In the future, an intriguing direction is to embed the multi-view procedure into the training process to jointly optimize both single-view and multi-view performance.

Chapter 6

Conclusion

The core challenge in visual understanding of object semantics and geometry is to obtain robust representations for different input modalities including RGB image, RGB-D data, point cloud and video sequence. Deep convolutional architectures are the current state-of-the-art feature learning machines for image classification. However, there are still two limiting factors that impedes their generalization performance. First, the spatially convolved and pooled features are sensitive to 3D object transformations (SE(3)), which leads to non-robust feature representations for parsing scene semantics from different viewpoints. Second, deep Convolutional Neural Networks, as end-to-end mapping machines, ignore inherent reasoning mechanism for a complex perception procedure so that they tend to overfit to training data with small size or generalize poorly between different domains (e.g. from synthetic to real data). In this dissertation, we have presented two methodologies which surmount above two limitations, and apply them as the guidelines to design the state-of-the-art learning architectures for large-scale classification, segmentation, 6-DoF pose estimation and 2D/3D keypoint localization of

rigid object instances. Additionally, we contribute four benchmarks for object instance classification and 6-DoF pose estimation.

In Chapter 3, we introduce the first methodology, called multi-domain pooling, for learning an object representation that is insensitive to 3D object transformations while preserving discriminative local details. We first formulate a probabilistic framework which reveals the following three perspectives:

- The pooling operation is essentially a variance reduction technique for input signals.
- The resolution of pooling regions (or pooling granularity) over a pooling space is correlated with the discrimination and invariance of the final pooled features. As the pooling granularity changes from fine to coarse levels, the pooled features obtain better robustness but less discrimination and vice versa. Statistically, the pooling granularity controls the trade-off between bias and variance of learned object features.
- A good discriminative representation can be learned by fine-grained pooling within domains that are insensitive to various object transformations. For example, the spatial domain is much less favorable than color domains in learning representation robust to 3D transformation, because the color signature of an object appearance changes much less than its spatial layout under an out-of-plane rotation.

Based on these three principles, we further propose a multi-scale and multi-domain pooling architecture that produces robust features for RGB image and 3D point cloud data. The key modules are first pooling local features

beyond the traditional spatial domain and then coupling small-scale local filters with fine-grained pooling scheme. We exploit additional pooling domains that are constructed by LAB color value, gradient SIFT [143] feature and local 3D geometry descriptor FPFH [60]. The joint space of these three domains is much more robust to 3D transformation and illumination than the traditional spatial domain. We present the state-of-the-art performance of multi-domain pooled features on large-scale instance recognition benchmarks including UW-RGBD [26], BigBIRD [27] and our own dataset JHUIT-50 [28]. Finally, we demonstrate another application of this object feature for instance segmentation by embedding it into a novel hierarchical semantic parsing framework.

Subsequently, we introduce the second methodology in Chapter 4. Visual perception often involves the sequential inference over a series of intermediate goals of growing complexity towards the final objective. For instance, knowing object orientation is a prerequisite to correctly infer object part visibility which in turn constrains the 3D locations of semantic object parts. Consequently, the overfitting occurs when a model follows an incorrect inference path that fits training data well, but generalizes poorly to the whole dataset. Motivated by this, we explore an approach for injecting prior domain structures into neural network training by supervising hidden layers of a CNN with intermediate concepts that normally are not observed in practice. We first employ a probabilistic framework to formalize the notions of intermediate concepts. This framework further points to better generalization through deep supervision, compared to the standard end-to-end training. This inspires

a CNN architecture where hidden layers are supervised with an intuitive sequence of intermediate concepts, in order to incrementally regularize the learning to follow the prescribed inference sequence. We practically leverage this superior generalization capability for complex single-view 3D structure prediction, where the model is required to generalize from synthetic CAD renderings to real images due to the scarcity of 3D annotations. The experiments demonstrate that our approach outperforms current state-of-the-art methods on 2D and 3D landmark prediction on public datasets, including KITTI-3D [28], PASCAL VOC, PASCAL3D [32] and IKEA [33]. In addition, we applied deep supervision to improve fine-grained image classification over single-task as well as multi-task networks on CIFAR100 [181]. Finally, we have presented preliminary results on jointly learning 3D geometry of multiple object classes within a single CNN.

Finally, we inspect the problem of estimating 6-DoF object pose in cluttered environments in Chapter 5. We start with a baseline system which first conducts the semantic segmentation based on multi-domain pooled features (Chapter 3) and then apply the off-the-shelf object registration technique ObjRecRANSAC [4] for pose estimation within each semantic partition of a given scene. To boost the single-view performance, we formulate a novel multi-view fusion framework which enables fast computation of multi-domain pooled features on an incrementally reconstructed 3D scene, as well as a robust aggregation scheme of semantic confidences returned from our single-view semantic segmentation algorithm. However, this baseline approach is still

limited in terms of accuracy and scalability because ObjRecRANSAC is sensitive to background clutter and not scalable to large numbers of object classes. Therefore, we present a new large-scale learning framework to predict the 6-DoF pose of multiple object classes in an end-to-end manner. We embed three novel modules into existing CNN architectures. First, we formulate a new pose representation of SE(3) based on a uniform tessellation of SO(3), which induces a robust inference scheme combining classification and regression. Next, we inject the prior of object classes into the development of convolutional filters, in order to learn class-specific pose features. Third, we borrow the deep supervision idea from Chapter 4 to regularize the training with an intermediate pose concept of object mask. Beyond the single view, we further introduce an multi-view algorithm that resolves the ambiguity of pose hypotheses from the CNN, based on a robust distance metric that takes object symmetry into account. We demonstrate the significant improvement of this proposed method against ObjRecRANSAC on JHUScene-50 and also the state-of-the-art performance on other public large-scale benchmarks including YCB-Video [17] and ObjectNet-3D [34].

6.1 Limitations and Future Work

Although we made advances to learning robust representations for various vision problems, there still exist some remaining challenges that should be addressed in future work.

Scalable Multi-Domain Pooling. Although our multi-domain pooling scheme achieves improvement over the current state-of-the-art over three

large-scale instance recognition benchmarks, two major limitations remain. First, fine-grained pooling in high levels (> 8) results in feature vectors with more than one million dimensions, though it is sparse via the soft-assignment encoder. This prevents more fine-grained implementations on large-scale data. Second, multiple domains are fused by concatenating feature descriptors of each domain. This may yield suboptimal performance of the final representation and is not scalable to large numbers of feature domains. To surmount the above two issues, one promising direction is to embed the pooling process into deep learning architecture and reduce the dimension of the concatenated feature vector via a feature encoder. In principle, the pooling operation is backpropagable regardless of pooling domain once we determine pooling regions. Thus, the local feature can be tuned to optimize its contribution to a specific pooling region in a pooling domain. Additionally, we might select a subset of representative pooling regions by using receptive field learning techniques [128, 129] to select a subset of representative pooling regions.

Multi-Class 3D Structure Learning. In Chapter 4, we evaluate our network, deeply supervised with intermediate shape concepts, mainly on car and three furniture object categories. One remaining question is how the current method can be scaled to hundreds or thousands of generic daily objects. There are two main difficulties on the way. First, we manually annotate those semantic keypoints before, which is clearly not a scalable method. However, it is non-trivial to efficiently obtain a fixed number of representative keypoints given an arbitrary CAD object model, especially when those object models are not well aligned within a canonical coordinate frame. One solution may

be finding a better 3D representation which can be easily computed given a CAD model and compactly describe large number of objects while being at fixed length for end-to-end training of CNN. Second, it seems that the current architecture is still limited to only being capable of learning features for a specific category or a small set of categories with similar shape characteristics (e.g. chair, sofa and bed). We might be able to resolve this by borrowing the idea of multi-class fusion module introduced in Section 5.3.2 to infer 3D structures of large number of classes.

More Applications using Deep Supervision. In this dissertation, we demonstrate three applications for the methodology of deep supervision with intermediate concepts, including 2D/3D keypoint localization for rigid objects, object pose estimation and image classification. In fact, this methodology can be applied in a broader range of scenarios. One simple extension is to estimate 3D structure of an deformable object such as human body. This is essentially a more complex problem than the rigid object setting typically in cluttered environments. It is worth trying the existing model to regress the keypoint locations for deformable objects. In addition, we also see wide applicability of deep supervision, even beyond computer vision, in domains such as robotic planning and scene physics inference [190], where inherent scene variables for a physics process can be potentially be used for deep supervision.

Incremental Multi-View Pose Estimation. In Section 5.3.3, we present a multi-view algorithm for selecting object pose hypotheses, in order to alleviate the problem of single-view ambiguity. Although we propose an efficient implementation which enables fast run-time performance, the time complexity

still grows linearly with the increasing number of views. As such, it becomes computationally expensive for a long sequence and is not applicable to an incremental setting where new observations continuously appear on-the-fly. One potential idea to solve this problem is to maintain a fixed set of "good" hypotheses and update it given a new frame. The other future direction is recognizing "key" views that are substantially different from each other and representative for a scene observation. In turn, we run a multi-view algorithm over these "key" views.

End-to-End Training of Multi-View Framework. Current multi-view pose estimation framework decouples single-view pose network training and multi-view fusion algorithm. One straightforward way to jointly optimize single-view CNN and multi-view selection is to exploit Recurrent Neural Networks (RNNs), such as LSTM. The major problem of this idea is the high model complexity due to redundancy. In the context of multi-view perception, the final prediction result is invariant to the input order, which is not favored by RNN like models because they are designed for learning temporal patterns. Thus, we need more parameters to learn the order-invariant feature. Moreover, different views should be registered into the same coordinate frame for fusion. It is not clear how to enforce the view alignment in the RNN framework. One alternative principled approach is to directly backpropagate the loss of pose hypotheses matching back to CNN and enable the end-to-end training. This induces several challenges. Technically, it is non-trivial to wrap up the multi-view procedures into a sequence of tensor operations. Further, we have to fix the number of views to construct an architecture with fixed size

for end-to-end training. Lastly, the gradient that is back-propagated into the single-view CNN may be vanished and numerically unstable.

In summary, deep architectures achieve substantial progress on parsing object semantics. However, they do not exploit scene geometry constraint to further boost the single-view results. In the future, the leaning machines may need to combine both semantics and geometry for the holistic understanding of the surrounding world, which not only serves for pure recognition purpose but more importantly leads to a deeper reasoning associated with complex human-scene interaction and task planning.

References

[1] Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. "Construction with quadrotor teams". In: *Autonomous Robots* 33.3 (2012), pp. 323–336.

[2] Jonathan Bohren et al. "A pilot study in vision-based augmented telemanipulation for remote assembly over high-latency networks". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3631–3638.

[3] Bertram Drost et al. "Model globally, match locally: Efficient and robust 3D object recognition". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. Ieee. 2010, pp. 998–1005.

[4] Chavdar Papazov and Darius Burschka. "An efficient ransac for 3d object recognition in noisy and occluded scenes". In: *Computer Vision–ACCV 2010*. 2011.

[5] Stefan Hinterstoisser et al. "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes". In: *Computer Vision–ACCV 2012*. Springer, 2013.

[6] Karl Pauwels et al. "Real-time object pose recognition and tracking with an imprecisely calibrated moving RGB-D camera". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 2733–2740.

[7] Nathanael Macias and John Wen. "Vision guided robotic block stacking". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 779–784.

[8] Scott Niekum et al. "Learning grounded finite-state representations from unstructured demonstrations". In: *The International Journal of Robotics Research* 34.2 (2015), pp. 131–157.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *NIPS*. 2012.

[10] Andreas Eitel et al. "Multimodal Deep Learning for Robust RGB-D Object Recognition". In: *IROS*. 2015.

[11] Max Schwarz, Hannes Schulz, and Sven Behnke. "RGB-D Object Recognition and Pose Estimation based on Pre-trained Convolutional Neural Network Features". In: (2015).

[12] Saurabh Gupta et al. "Learning rich features from RGB-D images for object detection and segmentation". In: *ECCV*. Springer, 2014.

[13] Nathan Silberman, David Sontag, and Rob Fergus. "Instance segmentation of indoor scenes using a coverage loss". In: *Computer Vision–ECCV 2014*. Springer, 2014.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *CVPR* (2015).

[15] Camille Couprie et al. "Indoor semantic segmentation using depth information". In: *arXiv preprint arXiv:1301.3572* (2013).

[16] Paul Wohlhart and Vincent Lepetit. "Learning descriptors for object recognition and 3d pose estimation". In: *CVPR*. 2015.

[17] Yu Xiang et al. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes". In: *arXiv preprint arXiv:1711.00199* (2017).

[18] Wadim Kehl et al. "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again". In: *CVPR*. 2017.

[19] Wadim Kehl et al. "Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation". In: *ECCV*. Springer. 2016, pp. 205–220.

[20] Vassileios Balntas et al. "Pose Guided RGBD Feature Learning for 3D Object Pose Estimation". In: *CVPR*. 2017.

[21] Shahram Izadi et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera". In: *ACM symposium on User interface software and technology*. ACM. 2011.

[22] Keisuke Tateno, Federico Tombari, and Nassir Navab. "Real-time and scalable incremental segmentation on dense SLAM". In: *IROS*. IEEE. 2015.

[23] Koray Kavukcuoglu et al. “Learning convolutional feature hierarchies for visual recognition”. In: *NIPS*. 2010.

[24] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “Multipath sparse coding using hierarchical matching pursuit”. In: *CVPR*. IEEE. 2013.

[25] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “Hierarchical matching pursuit for image classification: Architecture and fast algorithms”. In: *NIPS*. 2011.

[26] Kevin Lai et al. “A large-scale hierarchical multi-view rgb-d object dataset”. In: *ICRA*. 2011.

[27] Arjun Singh et al. “BigBIRD: A Large-Scale 3D Database of Object Instances”. In: *ICRA*. 2014.

[28] Chi Li, Austin Reiter, and Gregory D Hager. “Beyond Spatial Pooling: Fine-Grained Representation Learning in Multiple Domains”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4913–4922.

[29] Chi Li et al. “Hierarchical Semantic Parsing for Object Pose Estimation in Densely Cluttered Scenes. In:” in: *ICRA*. 2016.

[30] Chi Li et al. “Deep Supervision with Shape Concepts for Occlusion-Aware 3D Object Parsing”. In: *CVPR* (2017).

[31] Yi Yang and Deva Ramanan. “Articulated pose estimation with flexible mixtures-of-parts”. In: *CVPR*. 2011.

[32] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. “Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild”. In: *WACV*. 2014.

[33] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba. “Parsing IKEA Objects: Fine Pose Estimation”. In: *ICCV*. 2013.

[34] Yu Xiang et al. “ObjectNet3D: A Large Scale Database for 3D Object Recognition”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 160–176.

[35] Chen-Yu Lee et al. “Deeply-Supervised Nets”. In: *AISTATS* (2015).

[36] Chi Li et al. “Incremental scene understanding on dense SLAM”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 574–581.

[37] Chi Li, Jonathan Boheren, and Gregory D Hager. “Bridging the Robot Perception Gap With Mid-Level Vision. In:” in: *International Symposium on Robotics Research*. 2015.

[38] Aitor Aldoma et al. “A global hypotheses verification method for 3D object recognition”. In: *Computer Vision–ECCV 2012*. Springer, 2012, pp. 511–524.

[39] Kevin Lai et al. “Detection-based object labeling in 3D scenes”. In: *ICRA*. IEEE. 2012.

[40] Kevin Lai, Liefeng Bo, and Dieter Fox. “Unsupervised feature learning for 3d scene labeling”. In: *ICRA*. IEEE. 2014.

[41] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.

[42] Christian Szegedy et al. “Going deeper with convolutions”. In: CVPR. 2015.

[43] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv:1409.1556* (2014).

[44] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *JMLR* (2015).

[45] Kaiming He et al. “Deep residual learning for image recognition”. In: *CVPR* (2016).

[46] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CVPR’14* ().

[47] Ross Girshick. “Fast r-cnn”. In: *arXiv preprint arXiv:1504.08083* (2015).

[48] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *NIPS*. 2015.

[49] Kaiming He et al. “Mask r-cnn”. In: *ICCV*. IEEE. 2017.

[50] Daniel Wolf, Johann Prankl, and Markus Vincze. “Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015.

[51] Alexander Hermans, Georgios Floros, and Bastian Leibe. “Dense 3D semantic mapping of indoor scenes from RGB-D images”. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 2631–2638.

[52] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. “Perceptual organization and recognition of indoor scenes from RGB-D images”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 564–571.

[53] Umar Asif, Mohammed Bennamoun, and Ferdous Sohel. “Efficient RGB-D object categorization using cascaded ensembles of randomized decision trees”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 1295–1302.

[54] Henning Tjaden, Ulrich Schwanecke, and Elmar Schömer. “Real-Time Monocular Pose Estimation of 3D Objects using Temporally Consistent Local Color Histograms”. In: *CVPR*. 2017.

[55] Stefan Hinterstoisser et al. “Gradient response maps for real-time detection of textureless objects”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012).

[56] Alexander Krull et al. “Learning analysis-by-synthesis for 6d pose estimation in rgb-d images”. In: *ICCV*. 2015.

[57] Andy Zeng et al. “Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge”. In: *ICRA*. IEEE. 2017.

[58] Alykhan Tejani et al. “Latent-class hough forests for 3D object detection and pose estimation”. In: *ECCV*. Springer. 2014.

[59] S. Salti F. Tombari and L. Di Stefano. “A Combined Texture-Shape Descriptor For Enhanced 3D Feature Matching”. In: *ICIP* (2011).

[60] Radu Bogdan Rusu. “Semantic 3D object maps for everyday manipulation in human living environments”. In: *KI-Künstliche Intelligenz* 24.4 (2010), pp. 345–348.

[61] Eric Brachmann et al. “Learning 6d object pose estimation using 3d object coordinates”. In: *ECCV*. Springer, 2014.

[62] Eric Brachmann et al. “Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image”. In: *CVPR*. 2016.

[63] Frank Michel et al. “Global hypothesis generation for 6D object pose estimation”. In: *ICCV* (2017).

[64] Andreas Doumanoglou et al. “Recovering 6D object pose and predicting next-best-view in the crowd”. In: *CVPR*. 2016.

[65] Hao Su et al. “Render for CNN: Viewpoint estimation in images using CNNs trained with Rendered 3D model views”. In: *ICCV*. 2015.

[66] Francisco Massa, Renaud Marlet, and Mathieu Aubry. "Crafting a multi-task CNN for viewpoint estimation". In: *arXiv preprint arXiv:1609.03894* (2016).

[67] Arsalan Mousavian et al. "3d bounding box estimation using deep learning and geometry". In: *CVPR*. IEEE. 2017.

[68] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. "Real-Time Seamless Single Shot 6D Object Pose Prediction". In: *arXiv preprint arXiv:1711.08848* (2017).

[69] Mahdi Rad and Vincent Lepetit. "BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth". In: *ICCV*. 2017.

[70] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "Epnp: An accurate o (n) solution to the pnp problem". In: *International journal of computer vision* (2009).

[71] Tinghui Zhou et al. "Learning Dense Correspondence via 3D-guided Cycle Consistency". In: *CVPR'16* ().

[72] M Zeeshan Zia, Michael Stark, and Konrad Schindler. "Towards Scene Understanding with Detailed 3D Object Representations". In: *IJCV* (2015).

[73] Jiajun Wu et al. "Single Image 3D Interpreter Network". In: *ECCV'16* ().

[74] Tejas D Kulkarni et al. "Deep convolutional inverse graphics network". In: *NIPS*. 2015.

[75] Alexey Dosovitskiy, Jost Springenberg, and Thomas Brox. "Learning to Generate Chairs with Convolutional Neural Networks". In: *CVPR*. 2015.

[76] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Multi-view 3D Models from Single Images with a Convolutional Network". In: *ECCV*. 2016.

[77] Pol Moreno et al. "Overcoming Occlusion with Inverse Graphics." In: *ECCV*. 2016.

[78] Abhishek Kar et al. "Category-specific object reconstruction from a single image". In: *CVPR*. 2015.

[79] Christopher B Choy et al. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *ECCV*. 2016.

[80] Danilo Rezende et al. “Unsupervised Learning of 3D Structure from Images”. In: *NIPS'16* ().

[81] Yu Xiang et al. “Data-driven 3D voxel patterns for object category recognition”. In: *CVPR*. 2015.

[82] Mathieu Aubry et al. “Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models”. In: *CVPR*. 2014.

[83] Joseph J Lim, Aditya Khosla, and Antonio Torralba. “FPM: Fine pose Parts-based Model with 3D CAD models”. In: *ECCV*. 2014.

[84] Aayush Bansal, Bryan Russell, and Abhinav Gupta. “Marr revisited: 2D-3D Alignment via Surface Normal Prediction”. In: *CVPR*. 2016.

[85] Saurabh Gupta et al. “Inferring 3d object pose in RGB-D images”. In: *arXiv:1502.04652* (2015).

[86] Shubham Tulsiani and Jitendra Malik. “Viewpoints and Keypoints”. In: *CVPR'15* ().

[87] Xiang Yu, Feng Zhou, and Manmohan Chandraker. “Deep Deformation Network for Object Landmark Localization”. In: *ECCV* (2016).

[88] Angjoo Kanazawa, David W Jacobs, and Manmohan Chandraker. “WarpNet: Weakly Supervised Matching for Single-view Reconstruction”. In: *CVPR'16* ().

[89] Mingsheng Long et al. “Unsupervised domain adaptation with residual transfer networks”. In: *NIPS*. 2016.

[90] Baochen Sun, Jiashi Feng, and Kate Saenko. “Return of Frustratingly Easy Domain Adaptation”. In: *AAAI*. 2016.

[91] Philip Haeusser et al. “Associative Domain Adaptation”. In: *ICCV*. 2017.

[92] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. “An HOG-LBP Human Detector with Partial Occlusion Handling”. In: *ICCV*. 2009.

[93] Bojan Pepikj et al. “Occlusion Patterns for Object Class Detection”. In: *CVPR*. 2013.

[94] Tejas D Kulkarni et al. “Deep convolutional inverse graphics network”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 2539–2547.

[95] Alexander Thomas et al. “Towards multi-view object class detection”. In: *CVPR*. 2006.

[96] Alvaro Collet and Siddhartha S Srinivasa. “Efficient multi-view object recognition and full pose estimation”. In: *ICRA*. IEEE. 2010.

[97] Javier Civera et al. “Towards semantic SLAM using a monocular camera”. In: *IROS*. IEEE. 2011.

[98] Sudeep Pillai and John Leonard. “Monocular SLAM Supported Object Recognition”. In: *RSS*. 2015.

[99] Hang Su et al. “Multi-view convolutional neural networks for 3d shape recognition”. In: *CVPR*. 2015, pp. 945–953.

[100] Edward Johns, Stefan Leutenegger, and Andrew J Davison. “Pairwise decomposition of image sequences for active multi-view recognition”. In: *CVPR*. IEEE. 2016.

[101] Keisuke Tateno et al. “CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction”. In: *CVPR* (2017).

[102] Robert O Castle, Georg Klein, and David W Murray. “Combining monoSLAM with object recognition for scene augmentation using a wearable camera”. In: *Image and Vision Computing* (2010).

[103] Sid Yingze Bao and Silvio Savarese. “Semantic structure from motion”. In: *CVPR*. IEEE. 2011.

[104] Renato Salas-Moreno et al. “Slam++: Simultaneous localisation and mapping at the level of objects”. In: *CVPR*. 2013.

[105] Özgür Erkent, Dadhichi Shukla, and Justus Piater. “Integration of probabilistic pose estimates from multiple views”. In: *ECCV*. Springer. 2016.

[106] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories”. In: *CVPR*. IEEE. 2006.

[107] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “Unsupervised feature learning for rgb-d based object recognition”. In: *ISER* (2012).

[108] Richard Socher et al. “Convolutional-recursive deep learning for 3D object classification”. In: *NIPS*. 2012.

[109] Pierre Sermanet et al. “Overfeat: Integrated recognition, localization and detection using convolutional networks”. In: *arXiv preprint arXiv:1312.6229* (2013).

[110] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *ECCV*. 2014.

[111] Jeff Donahue et al. “Decaf: A deep convolutional activation feature for generic visual recognition”. In: *ICML* (2014).

[112] Thomas Berg and Peter N Belhumeur. “POOF: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation”. In: *CVPR*. IEEE. 2013.

[113] Yann LeCun, Fu Jie Huang, and Leon Bottou. “Learning methods for generic object recognition with invariance to pose and lighting”. In: *CVPR*. IEEE. 2004.

[114] Nicolas Pinto et al. “Comparing state-of-the-art visual features on invariant object recognition tasks”. In: *Applications of computer vision (WACV), 2011 IEEE workshop on*. IEEE. 2011.

[115] Ian Goodfellow et al. “Measuring invariances in deep networks”. In: *NIPS*. 2009.

[116] Qianli Liao, Joel Z Leibo, and Tomaso Poggio. “Learning invariant representations and applications to face verification”. In: *NIPS*. 2013.

[117] Yoshua Bengio and Yann LeCun. “Scaling learning algorithms towards AI”. In: *Large-scale kernel machines* (2007).

[118] Fabio Anselmi et al. “Unsupervised Learning of Invariant Representations in Hierarchical Architectures”. In: *arXiv preprint arXiv:1311.4158* (2013).

[119] Jim Mutch and David G Lowe. “Object class recognition and localization using sparse features with limited receptive fields”. In: *International Journal of Computer Vision (IJCV)* (2008).

[120] Koray Kavukcuoglu et al. “Learning invariant features through topographic filter maps”. In: *CVPR*. IEEE. 2009.

[121] Y-L Boureau et al. “Learning mid-level features for recognition”. In: *CVPR*. IEEE. 2010.

[122] Ken Chatfield et al. “Return of the Devil in the Details: Delving Deep into Convolutional Nets”. In: *arXiv preprint arXiv:1405.3531* (2014).

[123] Kihyuk Sohn and Honglak Lee. “Learning invariant representations with local transformations”. In: *ICML* (2012).

[124] Y-L Boureau et al. “Ask the locals: multi-way local pooling for image recognition”. In: *ICCV*. IEEE. 2011.

[125] Adam Coates and Andrew Y Ng. “Selecting receptive fields in deep networks”. In: *NIPS*. 2011.

[126] Sainbayar Sukhbaatar, Takaki Makino, and Kazuyuki Aihara. "Auto-pooling: Learning to Improve Invariance of Image Features from Image Sequences". In: *arXiv preprint arXiv:1301.3323* (2013).

[127] Sean Ryan Fanello et al. "Ask the image: supervised pooling to preserve feature locality". In: *CVPR*. IEEE. 2014.

[128] Mateusz Malinowski and Mario Fritz. "Learnable pooling regions for image classification". In: *arXiv preprint arXiv:1301.3516* (2013).

[129] Yangqing Jia, Chang Huang, and Trevor Darrell. "Beyond spatial pyramids: Receptive field learning for pooled image features". In: *CVPR*. IEEE. 2012.

[130] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Descriptor learning using convex optimisation". In: *ECCV*. Springer, 2012.

[131] Can Xu and Nuno Vasconcelos. "Learning Receptive Fields for Pooling from Tensors of Feature Response". In: *CVPR*. IEEE. 2014.

[132] Yunchao Gong et al. "Multi-scale orderless pooling of deep convolutional activation features". In: *ECCV*. Springer, 2014.

[133] Liujuan Cao et al. "Weakly supervised sparse coding with geometric consistency pooling". In: *CVPR*. IEEE. 2012.

[134] S. Salti F. Tombari and L. Di Stefano. "Unique Signatures of Histograms for Local Surface Description". In: *ECCV*. 2010.

[135] Walter Wohlkinger and Markus Vincze. "Ensemble of shape functions for 3D object classification". In: *Robotics and Biomimetics (ROBIO)*. IEEE. 2011.

[136] Aitor Aldoma et al. *OUR-CVFH–Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation*. 2012.

[137] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. "Kernel descriptors for visual recognition". In: *NIPS*. 2010.

[138] Manuel Blum et al. "A learned feature descriptor for object recognition in rgb-d data". In: *ICRA*. IEEE. 2012.

[139] Y-Lan Boureau, Jean Ponce, and Yann LeCun. "A theoretical analysis of feature pooling in visual recognition". In: *ICML*. 2010.

[140] Maximilian Riesenhuber and Tomaso Poggio. "Hierarchical models of object recognition in cortex". In: *Nature neuroscience* (1999).

[141] Manik Varma and Debajyoti Ray. “Learning the discriminative power-invariance trade-off”. In: *ICCV*. IEEE. 2007.

[142] Jan C Van Gemert et al. “Visual word ambiguity”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.7 (2010), pp. 1271–1283.

[143] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* (2004).

[144] Jan C Van Gemert et al. “Visual word ambiguity”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2010).

[145] Lingqiao Liu, Lei Wang, and Xinwang Liu. “In defense of soft-assignment coding”. In: *ICCV*. IEEE. 2011.

[146] Liefeng Bo et al. “Object recognition with hierarchical kernel descriptors”. In: *CVPR*. IEEE. 2011.

[147] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “Unsupervised feature learning for RGB-D based object recognition”. In: *Experimental Robotics*. Springer. 2013, pp. 387–402.

[148] Michael J Jones and Paul Viola. “Robust real-time object detection”. In: *Workshop on statistical and computational theories of vision*. Vol. 266. 2001, p. 56.

[149] Jeremie Papon et al. “Voxel cloud connectivity segmentation-supervoxels for point clouds”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013.

[150] Beverly J. Smith. “Perception of Organization in a Random Stimulus”. In: (1986).

[151] David Lowe. *Perceptual Organization and Visual Recognition*. Kluwer85.

[152] D. Marr and H. K. Nishihara. “Representation and recognition of the spatial organization of three-dimensional shapes”. In: *RSL B’78* ().

[153] R. Mohan and R. Nevatia. “Using perceptual organization to extract 3D structures”. In: *PAMI* (1989).

[154] S. Sarkar and P. Soundararajan. “Supervised learning of large perceptual organization”. In: *PAMI* (2000).

[155] Yaser S Abu-Mostafa. “Hints”. In: *Neural Computation* (1995).

[156] Rich Caruana. “Multitask learning”. In: Springer, 1998.

[157] Zhanpeng Zhang et al. “Facial landmark detection by deep multi-task learning”. In: *ECCV*. 2014.

[158] Jia Deng et al. “Large-Scale Object Classification Using Label Relation Graphs”. In: *ECCV*. 2014.

[159] J. Baxter. “A model of inductive bias learning”. In: *JAIR* (2000).

[160] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. “The benefit of multitask representation learning”. In: *JMLR* (2016).

[161] Çalar Gülçehre and Yoshua Bengio. “Knowledge matters: Importance of prior information for optimization”. In: *JMLR* (2016).

[162] Adrien Gaidon et al. “Virtual Worlds as Proxy for Multi-Object Tracking Analysis”. In: *CVPR*. 2016.

[163] Nikolaus Mayer et al. “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow and Scene Flow Estimation”. In: *CVPR*. 2016.

[164] German Ros et al. “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes”. In: *CVPR*. 2016.

[165] Henri Lebesgue. “Intégrale, longueur, aire”. In: *Annali di Matematica Pura ed Applicata (1898-1922)* 7.1 (1902), pp. 231–359.

[166] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* (1989).

[167] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. “Learning non-rigid 3d shape from 2d motion”. In: *NIPS*. 2003.

[168] Vikas Dhiman et al. “A Continuous Occlusion Model for Road Scene Understanding”. In: *CVPR*. 2016.

[169] Yu Xiang and Silvio Savarese. “Object Detection by 3D Aspectlets and Occlusion Reasoning”. In: *3dRR*. 2013.

[170] Tianfu Wu, Bo Li, and Song-Chun Zhu. “Learning And-Or Model to Represent Context and Occlusion for Car Detection and Viewpoint Estimation”. In: *PAMI* (2016).

[171] Hsi-Jian Lee and Zen Chen. “Determination of 3D human body postures from a single view”. In: *CVGIP* (1985).

[172] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *CVPR*. 2012.

[173] Angel X Chang et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv:1512.03012* (2015).

[174] Jianxiong Xiao et al. “Sun database: Large-scale scene recognition from abbey to zoo”. In: *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. 2010, pp. 3485–3492.

[175] Y. Li, L. Gu, and T. Kanade. “Robustly Aligning a Shape Model and Its Application to Car Alignment of Unknown Pose”. In: *TPAMI* (2011).

[176] Xiaowei Zhou et al. “Sparseness Meets Deepness: 3D Human Pose Estimation from Monocular Video”. In: *CVPR* (2016).

[177] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *AISTATS* (2010).

[178] Jonathan L Long, Ning Zhang, and Trevor Darrell. “Do convnets learn correspondence?” In: *NIPS*. 2014.

[179] Roozbeh Mottaghi, Yu Xiang, and Silvio Savarese. “A coarse-to-fine model for 3D pose estimation and sub-category recognition”. In: *CVPR*. 2015.

[180] Stephan R Richter et al. “Playing for data: Ground truth from computer games”. In: *ECCV*. 2016.

[181] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Technical Report, Chapter 3. University of Toronto, 2009.

[182] Dmytro Mishkin and Jiri Matas. “All you need is a good init”. In: *ICLR* (2016).

[183] Kaiming He et al. “Identity mappings in deep residual networks”. In: *ECCV’16* ().

[184] Federico Tombari and Luigi Di Stefano. “Object recognition in 3d scenes with occlusions and clutter by hough voting”. In: *Image and Video Technology (PSIVT), 2010 Fourth Pacific-Rim Symposium on*. IEEE. 2010, pp. 349–355.

[185] Maik Keller et al. “Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion”. In: *International Conference on 3D Vision (3DV)*. 2013.

[186] Eric Brachmann et al. “Learning 6d object pose estimation using 3d object coordinates”. In: *ECCV*. Springer. 2014.

[187] Gregory S Chirikjian et al. “Pose Changes From a Different Point of View”. In: *Journal of Mechanisms and Robotics* (2018).

[188] Yan Yan and Gregory S Chirikjian. "Almost-uniform sampling of rotations for conformational searches in Robotics and Structural Biology". In: *ICRA*. 2012.

[189] Sergey Levine et al. "Learning hand-eye coordination for robotic grasping with large-scale data collection". In: *International Symposium on Experimental Robotics*. Springer. 2016, pp. 173–184.

[190] Renqiao Zhang et al. "A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding". In: *arXiv preprint arXiv:1605.01138* (2016).

Vita

Chi Li is finishing his Ph.D. in Computer Science at Johns Hopkins University where he works on designing large-scale machine learning architectures for visual parsing object semantics and geometry. In 2012, he received his B.E. from Cognitive Science Department at Xiamen University (China), where he became interested in computer vision. His research mainly focuses on visual understanding of object properties from semantic class to 3D pose and structure. In particular, he is interested in leveraging scene geometry to enhance deep learning techniques on 2D/3D/Multi-view perception. He was part of the team which won the first prize in KUKA Innovation Award in 2016. Chi Li also gained industrial experiences from his three research internships with Apple, NEC Laboratories America and Microsoft Research.